Unit-5 - Planning and Expert System

Planning – planning problem – Simple planning agent – Blocks world problem – Mean Ends analysis Learning - Machine learning - Learning concepts, methods and models Introduction to expert system – architecture of expert systems.

Planning

Planning is a fundamental problem in artificial intelligence that involves finding a sequence of actions to achieve a goal.

Planning Problem:

A planning problem consists of:

1. **Initial state:** The starting state of the world.

2**. Goal state:** The desired state of the world.

3. **Actions:** A set of actions that can be taken to change the state of the world.

4. **Transition model:** A model that specifies the effects of each action.

Types of Planning:

1. **Classical planning:** Planning in a deterministic, fully observable environment.

2. **Probabilistic planning:** Planning in an environment with uncertainty.

3. **Planning under uncertainty:** Planning in an environment with incomplete knowledge.

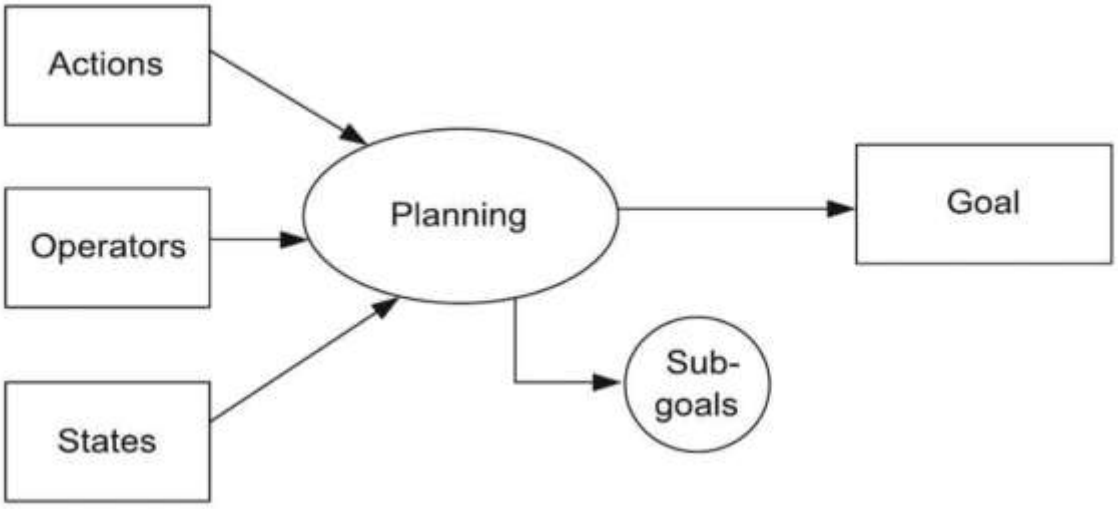
Planning Techniques:

1. **Forward planning:** Planning from the initial state to the goal state.

2**. Backward planning:** Planning from the goal state to the initial state.

3. **Heuristic search:** Using heuristics to guide the search for a plan.

Planning Representation



A basic planning representation typically includes:

1. States:

- A state represents the current situation or status of the world.

- States can be described using a set of variables or predicates.

2. Actions:

- An action is a transformation that changes the state of the world.

- Actions can be described using preconditions (what needs to be true before the action can be taken) and effects (what changes after the action is taken).

3. Goals:

- A goal is a desired state or situation.

- Goals can be represented using a set of predicates or constraints.

4. Plan:

- A plan is a sequence of actions that transforms the initial state into a goal state.

- Plans can be represented using a sequence of actions or a partial order plan.

Common Representations:

1. **STRIPS (Stanford Research Institute Problem Solver):** A classical planning representation that uses predicates and actions with preconditions and effects.

2. **PDDL (Planning Domain Definition Language):** A standardized language for representing planning domains and problems.

Components of planning

A planning system typically consists of several key components:

1. Planning Problem Definition:

- **Domain description:** A description of the planning domain, including the objects, actions, and constraints.

- **Problem instance:** A specific instance of the planning problem, including the initial state and goal.

2. Planning Algorithm:

- **Search algorithm:** An algorithm that searches for a plan, such as forward or backward search.

- **Heuristic function:** A function that estimates the distance from a state to the goal.

3. Plan Representation:

- **Plan structure:** A representation of the plan, such as a sequence of actions or a partial order plan.

- **Plan execution:** The process of executing the plan in the real world.

4. Knowledge Base:

- **Domain knowledge:** A knowledge base that contains information about the planning domain.

- **Action models:** Models of the actions that can be taken in the domain.

5. Planner:

- **Planner architecture:** The overall architecture of the planning system.

- **Planning engine:** The component that performs the planning.

6. Plan Evaluation:

- **Plan quality metrics:** Metrics used to evaluate the quality of the plan, such as cost or duration.

- **Plan validation:** The process of validating the plan to ensure it is correct and feasible.

Applications:

1**. Robotics:** Planning is used in robotics to navigate and manipulate objects.

2. **Logistics:** Planning is used in logistics to optimize routes and schedules.

3. **Game playing:** Planning is used in game playing to develop strategies.

Simple Planning Agent

A simple planning agent is a type of intelligent agent that uses planning to achieve its goals. Here's a basic outline:

Characteristics:

1. **Goal-oriented:** The agent has specific goals it wants to achieve.

2. **Planning:** The agent uses planning to generate a sequence of actions to achieve its goals.

3. **Action selection:** The agent selects actions based on the plan.

Components:

1. **Goal generator:** Generates goals for the agent.

2. **Planner:** Generates a plan to achieve the goals.

3**. Action selector:** Selects actions based on the plan.

Operation:

1. **Goal generation:** The agent generates a goal.

2. **Planning:** The agent generates a plan to achieve the goal.

3. **Plan execution:** The agent executes the plan.

Example:

A simple planning agent for a robot might have the goal of navigating to a specific location. The planner would generate a sequence of actions (e.g., move forward, turn left) to achieve the goal.

Benefits:

1. **Efficient problem-solving:** Planning agents can solve complex problems efficiently.

2. **Flexibility:** Planning agents can adapt to changing circumstances.

Limitations:

1**. Complexity:** Planning can be computationally expensive.

2. **Uncertainty:** Planning agents may struggle with uncertainty or incomplete knowledge.

Blocks world problem

The Blocks World problem involves a set of blocks with different colors and sizes, and a robotic arm that can manipulate the blocks. The goal is to move the blocks from an initial configuration to a goal configuration.

Key Features:

1**. Blocks:** The blocks have different colors and sizes.

2. **Robotic arm:** The robotic arm can pick up and move blocks.

3. **Initial configuration:** The initial arrangement of blocks.

4. **Goal configuration:** The desired arrangement of blocks.

Comments used:

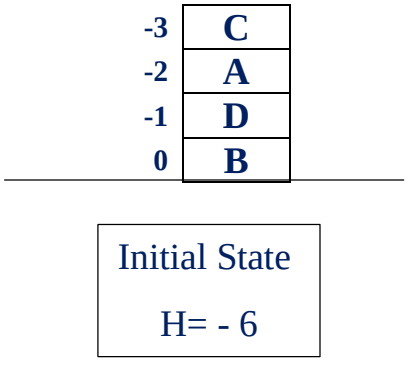
- **Unstack –** remove block from the stack
- **Stack –** add block in the stack
- **Putdown –** keep the block in the table
- **Pickup –** take the block from the table
- **Handempty** – Robot hand is empty.

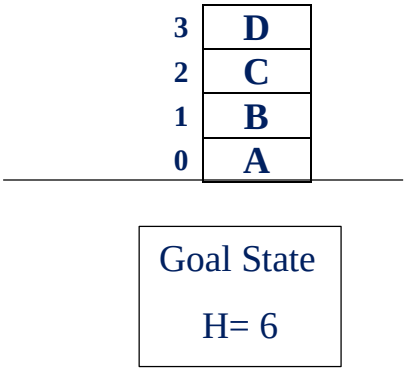
Example:

Initial state: ONTABLE (B), ON (D, B), ON (A, D), ON (C, A)

Goal State: ONTABLE (A), ON (B, A), ON(C, B), ON (D, C), CLEAR (D)

Convert the problem condition into Schematic representation:

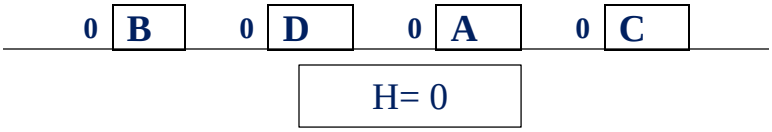




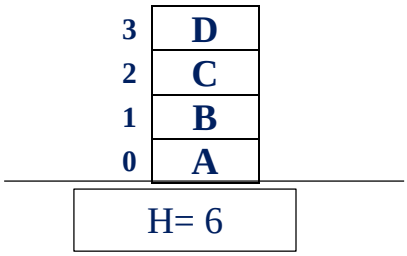
H –Heuristic Value

Algorithm

Unstack (C, A)
Putdown (C)
Unstack (A, D)
Putdown (A)
Unstack (D, B)
Putdown (D)



Pickup (B)
Stack (B, A)
Pickup (C)
Stack (C, B)
Pickup (D)
Stack (D, C)



Means-Ends Analysis

Means-Ends Analysis (MEA) is a problem-solving strategy used in artificial intelligence and cognitive science:

Key Concepts:

1. **Goal:** The desired state or outcome.

2. **Current state:** The current situation or status.

3. **Operators:** Actions or transformations that can be applied to the current state.

Means-Ends Analysis Process:

1. **Identify the goal:** Determine the desired state or outcome.

2. **Assess the current state:** Evaluate the current situation or status.

3. **Detect differences:** Identify the differences between the current state and the goal state.

4. **Apply operators:** Select and apply operators to reduce the differences between the current state and the goal state.

Applications:

1**. Artificial intelligence:** MEA is used in AI planning and problem-solving.

2**. Cognitive science:** MEA is used to model human problem-solving and decision-making.

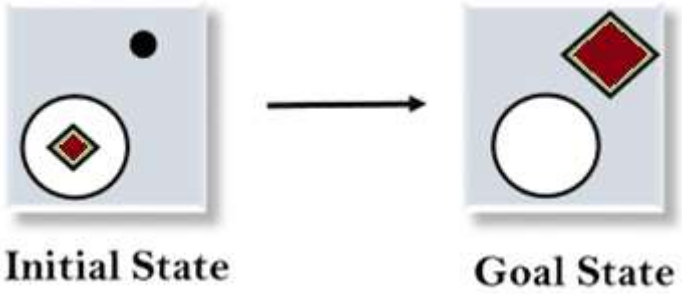
Benefits:

1. **Efficient problem-solving:** MEA can efficiently solve complex problems by focusing on the differences between the current state and the goal state.

2**. Flexibility:** MEA can be applied to a wide range of problems and domains.

Example:

Let's look at an example where we know the starting state and the desired state. In this issue, we must detect differences between the beginning state and the goal state and apply operators to obtain the goal state.



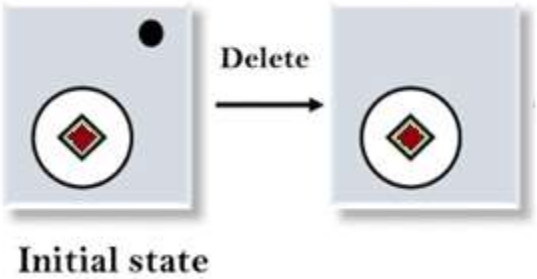
Solution

To solve the problem, we will first identify the differences between starting and goal states, then construct a new state and apply the operators to each difference. For this problem, we have the following operators:

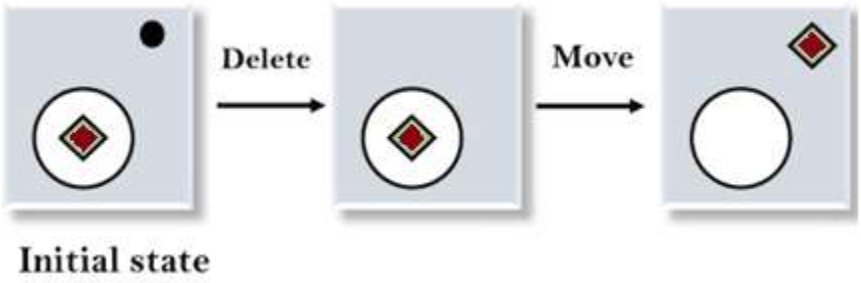
- Move
- Delete
- Expand

1. Evaluating the initial state: In the first step, we'll evaluate the initial state and compare it to the Goal state to see what the differences evaluating the initial state: In the first step, we'll evaluate the initial state and compare it to the Goal state to see what the differences are.

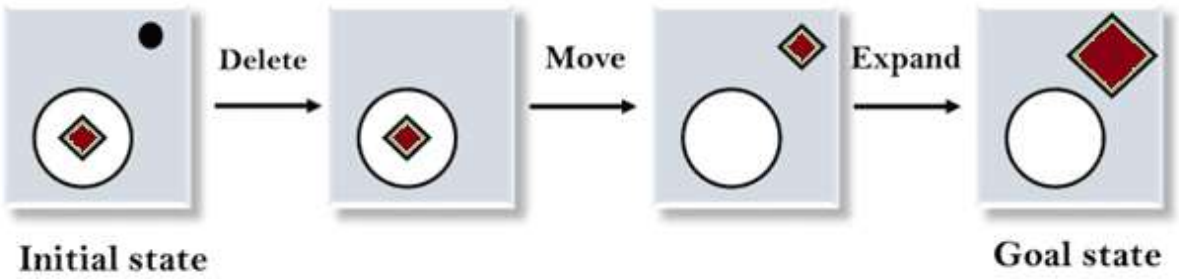
2. Applying the Delete operator: As you can see, the first difference is that there's no dot symbol in the Goal state, whereas there is in the initial state, so we'll use the Delete operator to remove it.



3. Applying the Move Operator: After using the Delete operator, a new state appears, which we will compare to the objective state again. After comparing these states, we notice that the square is outside the circle, so we'll use the Move Operator to fix it.



4. Applying the Expand Operator: In the third phase, a new state is created, and we will compare it to the desired state. There is still one difference between the states, which is the size of the square, so we will use the Expand operator to construct the desired state.



Machine learning

Machine learning is a subfield of artificial intelligence that involves training algorithms to learn from data and make predictions or decisions.

Learning Concepts:

1. **Supervised learning:** The algorithm learns from labeled data to make predictions.

2. **Unsupervised learning:** The algorithm learns from unlabeled data to identify patterns.

3. **Reinforcement learning:** The algorithm learns through trial and error by interacting with an environment.

Methods:

1. **Linear regression:** A linear model that predicts a continuous output variable.

2. **Decision trees:** A tree-based model that classifies data or makes predictions.

3. **Neural networks:** A model inspired by the human brain that can learn complex patterns.

4. **Clustering:** A method that groups similar data points together.

Models:

1. **Parametric models:** Models that assume a specific distribution of data (e.g., linear regression).

2. **Non-parametric models:** Models that don't assume a specific distribution of data (e.g., decision trees).

3. **Deep learning models:** Models that use multiple layers of neural networks to learn complex patterns.

Applications:

1. **Image classification:** Machine learning can be used to classify images into different categories.

2. **Natural language processing:** Machine learning can be used to analyze and generate human language.

3. **Recommendation systems:** Machine learning can be used to recommend products or services based on user behavior.

Benefits:

1. **Improved accuracy:** Machine learning can improve the accuracy of predictions and decisions.

2. **Automation:** Machine learning can automate tasks that would otherwise require human effort.

3. **Scalability:** Machine learning can handle large amounts of data and scale to meet the needs of complex applications.

Expert System

Expert systems are a crucial subset of artificial intelligence (AI) that simulate the decision-making ability of a human expert. These systems use a **knowledge base** filled with domain-specific information and rules to interpret and solve complex problems. For example, a medical expert system can **analyze a patient's symptoms and suggest possible diagnoses or treatments. Similarly, a financial expert system can evaluate market trends and recommend investment strategies.**

Why Are Expert Systems Important?

Expert systems are a game-changer in AI because they:

1. **Preserving Expertise**: They capture the knowledge of human experts and store it in a digital format. This ensures that valuable expertise isn't lost when an expert retires or leaves.
2. **Improving Decision-Making**: By relying on data and rules, expert systems provide consistent and unbiased recommendations.
3. **Saving Time and Money**: They automate tasks that would otherwise require human intervention, reducing costs and increasing efficiency.
4. **Accessibility**: Expert systems make expert-level knowledge available to non-experts, democratizing access to specialized information.

For instance, in the 1970s, the **MYCIN** system was developed to diagnose bacterial infections. While it was never used in real hospitals, it demonstrated how expert systems could assist doctors in making accurate diagnoses.

Components and Architecture of Expert System

An expert system is made up of several interconnected components, each playing a crucial role in its functionality. Let's break them down:

1. Knowledge Base: The Heart of the System

The knowledge base is the heart of an expert system. It contains all the facts, rules, and expert knowledge related to a specific domain. Think of it as a library filled with textbooks, research papers, and expert opinions. The ***accuracy and completeness of the knowledge base directly***

impact the system's performance. If the knowledge is outdated or incomplete, the system's recommendations may be flawed.

In a financial expert system, the knowledge base might include rules for detecting fraudulent transactions, such as "If a transaction exceeds $10,000 and occurs in a foreign country, flag it for review."

2. Inference Engine: The Brain behind the Decisions

The **inference engine** is the brain of the expert system. It processes the information stored in the knowledge base to draw conclusions or make recommendations. **The inference engine uses reasoning strategies (like forward chaining or backward chaining) to analyze data and apply rules.**

- **Forward Chaining**: Starts with available data and works toward a conclusion. For example, *"If the temperature is high and the patient has a cough, diagnose a respiratory infection."*
- **Backward Chaining**: Starts with a goal and works backward to find supporting evidence. For example, *"If the goal is to diagnose diabetes, check for symptoms like frequent urination and high blood sugar."*

3. User Interface: Bridging the Gap between System and User

The **user interface** is the bridge that allows users to interact with the expert system. It's designed to be intuitive and user-friendly, ensuring **that even non-experts can use the system effectively.** Users provide a **query** (problem or question), and the system processes the request. The system then delivers **advice** or recommendations back to the user.

4. Explanation Module: Building Trust through Transparency

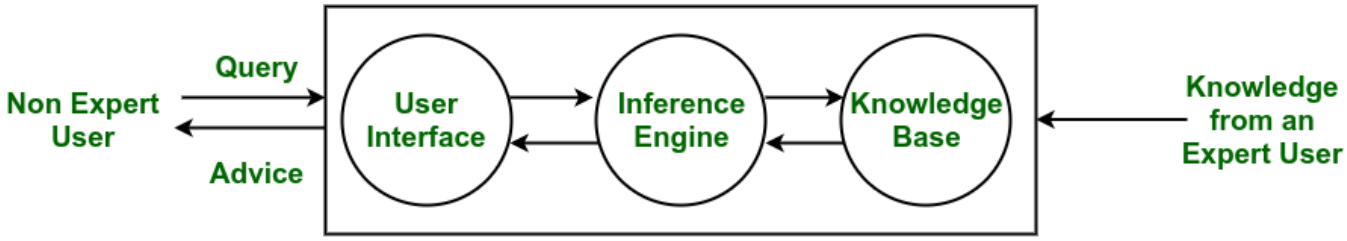
The **explanation module** is a critical feature that explains how the system arrived at a particular conclusion. It's like a teacher **showing their work when solving a math problem.** This module provides users with a clear, step-by-step explanation of the system's reasoning.

- This transparency is especially important in fields like healthcare and finance, where decisions can have significant consequences.
- **Example**: A medical expert system might explain, *"I diagnosed pneumonia because the patient has a fever, cough, and abnormal chest X-ray."*

5. Knowledge Acquisition Module: Keeping the System Up-to-Date

The **knowledge acquisition module** is responsible for updating and expanding the knowledge base. It ensures that the system stays current with the latest information and trends. Without regular updates, the system's knowledge base can become outdated, reducing its effectiveness.

Let's understand it's architecture with help of diagram:



The **working mechanism of an expert system** begins when a **non-expert user** submits a **query** through the **user interface**.

- This query is then processed by the **inference engine**, which applies **logical rules** and **reasoning techniques** to analyze the input.
- The **inference engine** interacts with the **knowledge base**, retrieving relevant **facts, rules, and heuristics** contributed by **expert users**.
- Based on this **structured knowledge**, the system derives **conclusions** and formulates an appropriate **response**.

Finally, the **expert system** provides **advice** or **recommendations** to the user, assisting in **decision-making** or **problem-solving** without requiring direct **human expert** intervention.

Types of Expert Systems in AI

Depending on their structure and application, expert systems can be categorized into different types.

1. Rule-Based Expert Systems

One of the most common types is **Rule-Based Expert Systems**, which rely on **if-then rules** to process information and make decisions. These rules are typically crafted by domain experts and serve as the system's reasoning mechanism. A well-known example is **MYCIN**, an early medical diagnosis system that identified bacterial infections.

2. Frame-Based Expert Systems

Another category is **Frame-Based Expert Systems**, which organize knowledge using **frames**, similar to objects in programming. These frames store attributes and values related to specific concepts, making them useful in natural language processing and other knowledge representation tasks.

3. Fuzzy Logic Systems

For situations involving **uncertainty and imprecision**, fuzzy logic Systems come into play. These systems don't operate on strict true/false values but instead allow for degrees of truth. **Fuzzy control systems**, commonly used in household appliances like **washing machines and air conditioners**, leverage this approach to optimize performance based on variable input conditions.

4. Neural Network-Based Expert Systems

Integrate artificial neural networks to **learn patterns from data** and improve decision-making. These systems are widely used in applications like **image recognition and speech processing**, where traditional rule-based approaches might struggle.

5. Neuro-Fuzzy Expert Systems

A more advanced hybrid approach is **Neuro-Fuzzy Expert Systems**, which merge the learning capabilities of neural networks with the uncertainty-handling strengths of **fuzzy logic**. These systems are particularly useful in **financial forecasting** and **automated control systems**, where both structured learning and flexible reasoning are necessary.

Examples of Expert Systems in AI

There have been several significant real-world expert systems developed over the years. Some of them are given below:

1. MYCIN : As mentioned earlier, revolutionized medical diagnosis by using rule-based logic to detect bacterial infections.

- MYCIN uses **backward chaining** to diagnose bacterial infections, such as meningitis and bacteremia. It identifies the bacteria causing the infection by asking the doctor a series of questions about the patient's symptoms and test results.
- **Significance**: Although not used clinically, MYCIN greatly influenced the development of medical expert systems.

2. DENDRAL : One of the earliest AI systems in chemistry, could analyze mass spectrometry data to predict molecular structures.

- DENDRAL was designed to analyze chemical compounds. It uses **spectrographic data** (data obtained from spectroscopy) to predict the molecular structure of a substance.
- **Significance**: DENDRAL revolutionized chemical research by automating the analysis of mass spectrometry data.

3. R1/XCON: R1, also known as XCON, was developed in the late 1970s by Digital Equipment Corporation (DEC) and is one of the most commercially successful expert systems.

- R1/XCON was used to configure orders for new computer systems. It would select the appropriate hardware and software components based on the customer's requirements.
- **Significance**: R1/XCON streamlined system configuration, saving DEC millions by reducing errors and improving efficiency.

4. PXDES: PXDES is an expert system designed for the medical field, particularly in the diagnosis of lung cancer.

- PXDES could analyze patient data, including imaging results, to determine both the type and the stage of lung cancer. It helps in deciding the best course of treatment based on the patient's specific condition.
- **Significance:** PXDES aids in accurate, timely diagnoses, improving treatment decisions in oncology.

5. CaDet: CaDet is a clinical support system developed to assist in the early detection of cancer.

- CaDet can identify potential signs of cancer in its early stages by analyzing patient data and symptoms. It works by comparing patient data with known patterns and indicators of cancer.
- **Significance**: Early detection by CaDet enhances survival rates by enabling prompt treatment.

6. DXplain: DXplain is a medical expert system developed at Massachusetts General Hospital, used as a clinical decision support tool.

- DXplain suggests possible diseases based on the symptoms and findings provided by a doctor. It acts as a reference tool, offering a differential diagnosis list that doctors can use to check their own diagnoses.
- **Significance:** DXplain broadens diagnostic possibilities, helping medical professionals consider rare conditions.

Applications of Expert Systems

1. **Medical Diagnosis**: Expert systems assist doctors by analyzing symptoms and medical history to suggest possible diagnoses or treatment options. For example, MYCIN, an early expert system, helped identify bacterial infections and recommend antibiotics.
2. **Financial Services**: In finance, expert systems are used for credit scoring, fraud detection, and investment advice. They analyze financial data and patterns to make informed decisions.
3. **Technical Support**: Expert systems can troubleshoot and provide solutions for technical issues. They guide users through problem-solving steps based on pre-defined rules and knowledge.
4. **Manufacturing**: In manufacturing, expert systems help optimize production processes, perform quality control, and manage inventory by analyzing data and making recommendations.

Benefits of Expert Systems

1. **Consistency**: Expert systems provide consistent and reliable recommendations, reducing the variability that can occur with human decision-making.
2. **Availability**: They are available 24/7 and can handle multiple queries simultaneously, providing timely assistance and support.

3. **Cost-Effectiveness**: By automating expert-level decision-making, organizations can save on the costs associated with hiring and training human experts.

4. **Knowledge Preservation**: Expert systems preserve valuable knowledge and expertise, making it accessible even if the original experts are no longer available.

Limitations of Expert Systems

1. **Knowledge Limitation**: The effectiveness of an expert system depends on the completeness and accuracy of the knowledge base. If the knowledge is outdated or incomplete, the system's performance may be compromised.

2. **Lack of Flexibility**: Expert systems are limited to the rules and knowledge they are programmed with. They may struggle with novel or ambiguous situations that fall outside their predefined rules.

3. **Maintenance**: Regular updates and maintenance are required to keep the knowledge base current and relevant, which can be resource-intensive.