UNIT 2

Chapter 3

Knowledge Representation

Knowledge Representation (KR) is a subfield of artificial intelligence (AI) that focuses on how to formally represent information about the world in a format that a computer system can utilize to solve complex tasks such as diagnosing a problem, making decisions, or understanding natural language. KR seeks to encode knowledge in such a way that it enables machines to simulate human understanding, reasoning, and decision-making processes.

Importance of Knowledge Representation

1. **Facilitates Understanding**: Knowledge representation allows machines to understand and interpret human knowledge, making it possible for them to interact meaningfully with people.
2. **Enables Reasoning**: By representing knowledge, machines can perform logical reasoning, derive new information, and make inferences based on existing knowledge.
3. **Supports Learning**: Knowledge representation provides a framework for machine learning algorithms to structure and learn from data effectively.
4. **Enhances Problem Solving**: Proper representation of knowledge allows for efficient problem-solving techniques in various applications, such as expert systems, natural language processing, and robotics.

Types of Knowledge Representation

1. **Semantic Networks**: Graph structures that represent knowledge in terms of nodes (concepts) and edges (relationships), facilitating the representation of hierarchies and associations.
2. **Frames**: Data structures that hold information about an object or concept, including attributes and values, akin to objects in object-oriented programming.
3. **Production Rules**: If-Then rules that dictate actions based on certain conditions, often used in expert systems.
4. **Logic-Based Representations**: Using formal logic, such as propositional or first-order logic, to represent knowledge. This allows for rigorous reasoning but can become complex with larger datasets.
5. **Ontologies**: Formal representations of a set of concepts within a domain, including the relationships between them. Ontologies help standardize vocabulary and semantics.

6. **Bayesian Networks**: Probabilistic graphical models that represent a set of variables and their conditional dependencies, allowing for reasoning under uncertainty.

Issues in Knowledge Representation

Despite its significance, knowledge representation faces several challenges:

1. **Complexity of Real-World Knowledge**:Real-world knowledge is often complex, dynamic, and multifaceted. Representing this complexity in a structured format can be daunting. Many systems struggle to capture the full range of knowledge necessary for comprehensive understanding and reasoning.

2. **Ambiguity and Vagueness**: Natural language is often ambiguous and vague, making it difficult to represent knowledge in a precise and unambiguous manner. Different interpretations of the same information can lead to misunderstandings and incorrect conclusions.

3. **Scalability**: As the amount of knowledge increases, maintaining and processing that knowledge becomes increasingly challenging. Efficiently scaling knowledge representation systems to handle large datasets while ensuring fast retrieval and reasoning is a significant issue.

4. **Expressiveness vs. Computational Efficiency**:There is often a trade-off between the expressiveness of a representation (how much detail and nuance it can capture) and its computational efficiency (how quickly a computer can process and reason with it). Highly expressive representations may lead to slower processing times.

5. **Subjectivity in Knowledge Acquisition**: Knowledge representation often relies on human input, which can introduce biases and inconsistencies. The process of acquiring and validating knowledge can be subjective and error-prone, impacting the quality of the knowledge base.

6. **Incompleteness and Uncertainty**: Knowledge can be incomplete or uncertain, and traditional representation methods may not adequately capture these aspects. This is especially true in domains like healthcare or finance, where information is often uncertain or incomplete.

7. **Interoperability**: Different systems may use different knowledge representation formats, leading to challenges in sharing and integrating knowledge across platforms. Achieving interoperability among diverse systems is a significant hurdle.

8. **Maintenance and Update Issues**: Knowledge bases need to be regularly updated to reflect new information or changes in the domain. Managing these updates and ensuring consistency can be complex and resource-intensive.

9. **Human-Centric Representation**: Many knowledge representation systems are designed based on human understanding, which may not always align with how machines process information. Bridging this gap is crucial for effective AI applications.

Semantic Networks

Semantic networks are a form of knowledge representation that utilizes a graphical structure to represent knowledge in a way that highlights the relationships between concepts. They are particularly useful in artificial intelligence, natural language processing, and cognitive science for modeling how information is organized and how entities are related.

Key Concepts of Semantic Networks

1. **Nodes**: Nodes represent concepts, entities, or objects within the semantic network. For example, in a network about animals, nodes might represent "Dog," "Cat," "Animal," etc.
2. **Links (Edges)**:Links connect nodes and represent the relationships between them. These relationships can vary in type and nature, such as hierarchical, associative, or part-whole relationships.
3. **Types of Relationships**:
    - **Is-a Relationship**: Represents a hierarchical relationship where one node is a subtype of another. For example, "Dog" is a type of "Animal."
    - **Has-a Relationship**: Represents a part-whole relationship. For example, "Car" has "Wheels."
    - **Related-to Relationship**: Represents an associative relationship that connects two nodes that are related in some way but do not fit into the is-a or has-a categories. For example, "Teacher" is related to "Student."

Structure of Semantic Networks

Semantic networks can be represented visually as graphs, where:

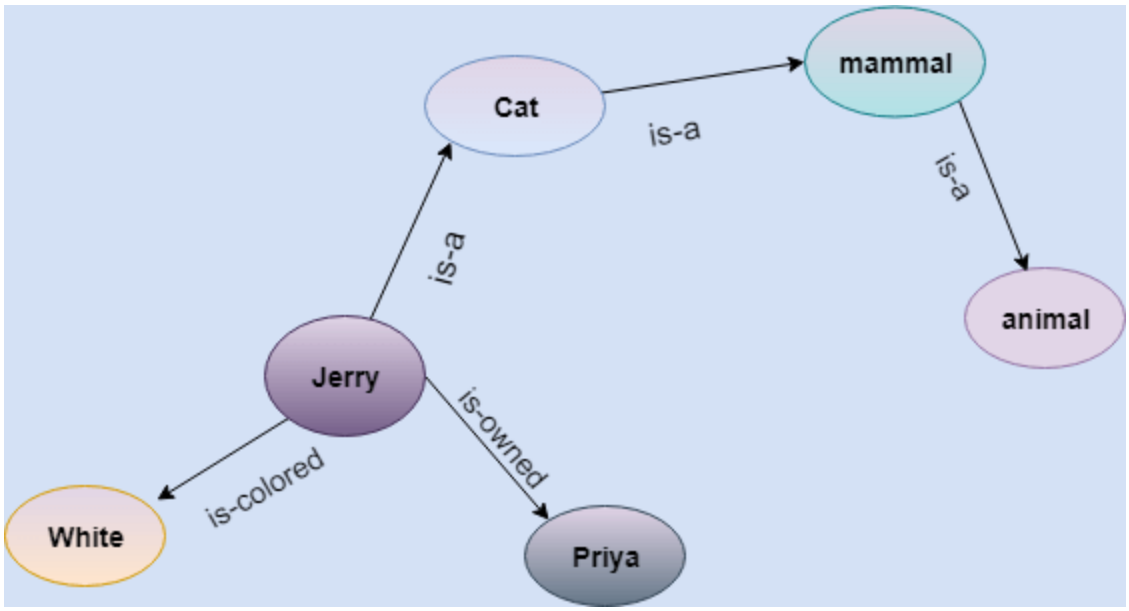
- Nodes are represented as circles or ovals.
- Links are represented as lines connecting the nodes.

- Different types of relationships can be represented using different styles of lines or labels on the lines.

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.
4. Jerry is brown colored.
5. All Mammals are animal.



Types of Semantic Networks

1. **Directed Semantic Networks**: The relationships are represented with directed edges (arrows) to indicate the direction of the relationship. For example, "A is a type of B" is shown as an arrow pointing from A to B.
2. **Undirected Semantic Networks**:Relationships are represented with undirected edges, indicating a bidirectional relationship between the nodes.
3. **Weighted Semantic Networks**:Links between nodes can have weights, representing the strength or relevance of the relationship. This is often used in information retrieval systems.

Advantages of Semantic Networks

1. **Visual Representation**: Semantic networks provide a clear and intuitive visual representation of knowledge, making it easier to understand relationships among concepts.
2. **Flexibility**: They can represent various types of relationships and can be easily expanded to include new nodes and relationships.
3. **Inference Capability**: Semantic networks can support reasoning and inference by allowing systems to draw conclusions based on the relationships between concepts.
4. **Natural Language Processing**: They are useful in understanding and processing natural language by representing the meanings of words and their interrelations.
5. **Cognitive Modeling**:Semantic networks can model human knowledge and cognition, providing insights into how people organize and retrieve information.

Limitations of Semantic Networks

1. **Ambiguity**: The meaning of relationships can be ambiguous, leading to potential misinterpretations if not clearly defined.
2. **Scalability**: As the number of nodes and relationships increases, the complexity of the network can grow significantly, making it harder to manage and interpret.
3. **Lack of Formal Semantics**: Semantic networks may lack rigorous formal semantics, leading to challenges in automated reasoning and inference.
4. **Limited Expressiveness**: They may not be able to express certain types of knowledge or complex relationships effectively compared to other representation methods like ontologies or frames.

Applications of Semantic Networks

1. **Artificial Intelligence**: Used for knowledge representation in expert systems, allowing for automated reasoning and decision-making.
2. **Natural Language Processing (NLP)**: Facilitates tasks such as word sense disambiguation, information retrieval, and sentiment analysis.
3. **Cognitive Science**: Models human knowledge structures and cognitive processes, providing insights into how people organize and recall information.
4. **Information Retrieval**: Enhances search engines and databases by representing concepts and their relationships, improving the accuracy of search results.
5. **Ontology Development**: Semantic networks can serve as a foundation for developing more formal ontologies in various domains, such as biomedical and social sciences.

Frames

Frames are one of the most widely used structures for knowledge representation in Artificial Intelligence (AI). They were introduced by Marvin Minsky in the 1970s as a way to represent stereotypical situations, objects, or events in a structured and easily interpretable format. A frame represents a collection of attributes or properties related to an entity, object, or concept, along with their corresponding values. Frames allow for hierarchical and structured knowledge storage, enabling machines to understand and infer information about the real world.

Components of a Frame Structure

1. **Frame Name:** The name or identifier of the frame that represents a concept, object, or event. The frame name serves as a label for what the frame is describing (e.g., a person, a vehicle, an event).
2. **Slots (Attributes):**
    - Slots define the attributes or properties of the object or event being represented by the frame. They describe the different characteristics or aspects of the concept, much like fields in a database or properties of an object in object-oriented programming.
    - Each slot can hold a value, a set of values, a default value, or even a procedure that determines its value.
3. **Fillers (Values)/Filters:**
    - The fillers represent the actual data or values assigned to each slot. For example, if the slot is Name, the filler could be "John". Fillers can be constants, other frames, lists, or even computed values.
    - Fillers can also take default values when no explicit value is provided, which is especially useful when frames are used as templates for multiple objects.
4. **Default Values:**
    - Default values are provided for slots when specific information is not available. For example, a frame for "Bird" may have a default value of canFly = True. This saves time when populating frames for similar objects, as many instances will share the same default values.
5. **Inheritance:**
    - Frames are often organized hierarchically, where a frame can inherit properties from a parent or more general frame. Inheritance allows for reuse and organization of common attributes.
    - For instance, a frame for "Car" might inherit from a more general frame called "Vehicle," meaning the "Car" frame will automatically include all the slots and

values of the "Vehicle" frame (such as Wheels = 4), while adding specific details about the car (like Brand and Model).

6. **Procedural Attachments:**
   - Procedural attachments are mechanisms or methods attached to slots that dynamically calculate or infer the slot's value based on some conditions or procedures. For example, a frame representing a person's age could have a procedure that calculates the age based on the birth date and the current date.
7. **Conditions and Constraints:**
   - Slots can also contain conditions or constraints that define what values are acceptable or what actions should be taken if certain criteria are met. For example, a slot for Speed in a vehicle frame might have a constraint that limits the value to a range (e.g., between 0 and 250 km/h).

Frame Structure: Example

Let's consider an example frame for representing a Person in knowledge representation. Here's what the structure might look like:

Frame Name: "Person"

Slot (Attribute)	Filler (Value)	Default Value	Type	Inheritance	Procedure / Condition
Name	"John Doe"	None	Text	No	None
Age	30	Unknown	Integer	No	Derived from birth date
Gender	Male	None	Text	No	None
Occupation	Engineer	Unknown	Text	No	None
Address	"123 Main St, City"	None	Text	No	None
Birthdate	"January 1, 1990"	None	Date	No	None
Marital Status	Single	Unknown	Text	No	None

Explanation of the Structure:

- **Frame Name ("Person"):** This is the label for the frame. It identifies that this frame is representing a person.
- **Slots (Attributes):** Attributes like Name, Age, Gender, Occupation, and Address represent the characteristics of the person.
- **Fillers (Values):** The fillers hold the actual values for each slot. For example, the Name is "John Doe", and the Age is 30.
- **Default Values:** If specific information is not provided, some slots may have default values. For example, the slot Occupation has a default value of Unknown if the person's occupation is not specified.
- **Type:** The type of each slot is specified (e.g., Text, Integer, Date) to define what kind of data the slot holds.
- **Inheritance:** In this example, there is no inheritance. But if the "Person" frame inherited from a more general frame like "Living Being," it could automatically inherit attributes such as Species = Human.
- **Procedures:** For example, the Age slot could have a procedure attached that calculates the current age by subtracting the Birth date from the current date.

Advantages of Frames in Knowledge Representation

1. **Modularity**: Frames provide a modular way to represent knowledge by dividing information into manageable and reusable parts.
2. **Hierarchical Structure**: Frames allow for easy inheritance of properties, supporting the reuse of common attributes and ensuring consistency across related objects.
3. **Default Values**: The ability to assign default values to slots allows for flexibility, as missing information can be inferred from the default.
4. **Procedural Attachments**: Frames can dynamically compute values, making them adaptable to changing situations or contexts.
5. **Real-World Modeling**: Frames are well-suited for representing real-world knowledge, particularly in domains with structured and semi-structured data, such as expert systems and natural language processing.

Issues and Limitations of Frames

1. **Lack of Expressiveness for Complex Relationships**: Frames are not always suitable for representing complex relationships between entities, such as those requiring logical or temporal reasoning.
2. **Handling Exceptions**: Frames typically operate on generalizations, which can make it difficult to handle exceptions. For example, most birds can fly, but representing non-flying birds like penguins can be cumbersome.

3. **Ambiguity**: Frames may struggle with ambiguous or incomplete data, especially in dynamic environments where the exact attributes or values may change over time.
4. **Inference Limitations**: Although frames provide a good structure for representing knowledge, they are less powerful in terms of inference compared to rule-based or logic-based systems.
5. **Scalability**: In large systems, managing an extensive hierarchy of frames can become complex, and updating knowledge across frames may lead to inconsistencies.

Inheritance

Inheritance in knowledge representation refers to the ability of objects or concepts to acquire properties, behaviors, or relationships from more general classes or categories. This mechanism is commonly used in artificial intelligence (AI) and computer science, particularly in systems that organize and represent knowledge hierarchically.

Key Concepts:

1. **Class Hierarchies (Taxonomies):**
    - Inheritance organizes knowledge in a hierarchy of classes and subclasses.
    - Higher-level classes (or *parent classes*) represent general concepts, while lower-level classes (*child classes*) represent more specific concepts.
    - Example: In a hierarchy of animals, "Mammal" is a parent class, and "Dog" is a child class. A dog inherits properties of mammals (e.g., warm-blooded).
2. **Property Inheritance:**
    - Child classes inherit properties and attributes from their parent classes.
    - Example: If the parent class "Bird" has a property "can fly," its subclasses like "Eagle" and "Sparrow" will inherit this property, unless explicitly overridden (e.g., "Penguin" may not inherit "can fly").
3. **Default Reasoning:**
    - Inheritance allows for default reasoning, where specific instances inherit default values from their class unless there's an exception.
    - Example: Most birds can fly, so a subclass like "Eagle" inherits this property, but "Penguin" can override it with "cannot fly."
4. **Multiple Inheritance:**
    - Some knowledge representation systems allow multiple inheritance, where a class can inherit from more than one parent.
    - Example: A "Flying Fish" could inherit properties from both "Fish" and "Flying Animal" classes.
5. **Advantages of Inheritance in Knowledge Representation:**
    - **Reusability:** Common properties and behaviors can be defined once and reused across multiple subclasses, reducing redundancy.

- **Abstraction:** Higher-level concepts are abstracted to capture general knowledge, while details are refined at lower levels.
- **Efficiency:** It allows for more efficient reasoning, as properties don't need to be redefined for every specific instance.

Example in AI Knowledge Representation:

In a semantic network, inheritance helps establish relationships:

- Class: "Vehicle"
  - Property: has wheels
  - Subclass: "Car"
    - Inherits "has wheels" from "Vehicle"
    - Adds new properties like "has steering wheel"
  - Subclass: "Bicycle"
    - Inherits "has wheels" from "Vehicle"
    - Adds properties like "has pedals"

Difference between Semantic Networks and Frames

Semantic networks and frames are both used for knowledge representation but differ in their structure and approach:

Aspect	Semantic Networks	Frames
Representation	Graphical representation with nodes and edges.	Data structures with slots and values.
Components	Nodes (concepts), edges (relationships).	Slots (attributes) and values (fillers).
Structure	Hierarchical or associative network of concepts.	Structured, often hierarchical, but focused on specific entities or scenarios.
Purpose	Represents relationships between concepts, often used for reasoning.	Represents stereotypical knowledge about objects or situations.
Usage	Used in knowledge representation,	Used to model structured

Aspect	Semantic Networks	Frames
	reasoning, and inference.	knowledge, like objects, events, or scenarios.
Flexibility	More flexible in representing complex relationships.	More rigid, with predefined slots and values.
Example	"Dog is a Mammal" and "Mammal is an Animal" relationship.	A frame for "Car" might have slots for "Make," "Model," "Color."
Handling of Default Values	Does not inherently handle default values.	Can include default values for slots, which can be overridden.
Reasoning Capability	Supports inference through relationships (e.g., inheritance).	Typically involves simple procedural attachments for reasoning.
Ease of Modification	Can be more complex to modify due to interconnected relationships.	Easier to modify specific frames by adjusting slots and values.

Constraint propagation

Constraint propagation is a technique used in **constraint satisfaction problems (CSPs)** to reduce the search space by systematically enforcing constraints on the variables in a problem. It involves deducing and propagating the effects of constraints, which helps to simplify the problem by eliminating inconsistent or invalid variable assignments early in the search process.

Key Concepts of Constraint Propagation:

1. **Constraint Satisfaction Problems (CSPs):**
    - A CSP consists of variables, domains (possible values for each variable), and constraints (rules that must be satisfied).
    - The goal is to assign values to variables such that all constraints are met.

- Examples of CSPs include scheduling, Sudoku puzzles, and graph coloring problems.

2. **Propagation of Constraints:**
   - Constraint propagation works by applying constraints to narrow down the domain of possible values for each variable.
   - As the domains are reduced, the narrowed options may further reduce the domains of other variables through **propagation** of the constraints, hence minimizing the potential combinations to explore.
3. **Example of Constraint Propagation:**
   - Consider a simple Sudoku puzzle. If a number "3" is placed in a cell in one row, the constraint that no row can have duplicate numbers is applied, so the number "3" can be removed from the possible choices in the other cells of that row. This change propagates further constraints to other rows, columns, and grids.
4. **Arc Consistency:**
   - A common form of constraint propagation is **arc consistency**. A variable is arc-consistent with another if, for every value in the domain of the first variable, there is some valid value in the domain of the second variable that satisfies the constraint between them.
   - For example, in graph coloring, if two adjacent nodes must have different colors, constraint propagation ensures that their color domains are consistently reduced.
5. **Techniques in Constraint Propagation:**
   - **Forward Checking:** When assigning a value to a variable, forward checking immediately checks if this value eliminates all possible values for any future variable. If so, it backtracks.
   - **AC-3 Algorithm:** An algorithm used to achieve arc consistency by removing values that do not satisfy constraints between variables.
   - **Generalized Constraint Propagation:** Can be extended to higher-order constraints where relationships exist among multiple variables, rather than just pairs.
6. **Advantages of Constraint Propagation:**
   - **Pruning the Search Space:** By reducing the domain of variables early, constraint propagation minimizes unnecessary search, speeding up problem-solving.
   - **Increased Efficiency:** It makes CSP solvers more efficient, as fewer possibilities need to be explored after propagation.

- **Backtracking Reduction:** By eliminating invalid variable assignments early, constraint propagation reduces the number of times the algorithm needs to backtrack due to conflicts.

Applications of Constraint Propagation:

- **Sudoku Solving:** By propagating constraints, a solver can prune incorrect numbers from the grid faster.
- **Scheduling Problems:** In tasks where resources or time slots are limited, constraint propagation helps eliminate invalid scheduling options.
- **Map Coloring/Graph Coloring:** Reducing the domain of colors for each node while ensuring adjacent nodes have different colors.
- **Cryptarithmetic Puzzles:** Letters in puzzles must represent different digits, and constraint propagation helps enforce such restrictions.

Rule Based Knowledge Systems

Knowledge representation is one of the core components of Artificial Intelligence (AI). One powerful method of representing knowledge is through **rules**, which consist of logical statements that dictate how knowledge is structured and used for reasoning. Rule-based systems are used in expert systems, decision-making processes, and various AI applications where reasoning about specific conditions or facts is essential.

Types of Rules in Knowledge Representation

1. **Production Rules (If-Then Rules):** These are the most common type of rules used in rule-based systems. They are composed of two parts:
   - **Antecedent (Condition/If part):** This specifies the conditions that need to be satisfied for the rule to be applied.
   - **Consequent (Action/Then part):** This defines the action or conclusion that follows if the conditions are satisfied.

   A production rule can be written as:

   IF condition(s) THEN action(s)/conclusion

   Example: IF the light is red THEN stop the car.

   Production rules are commonly used in decision-making systems such as expert systems, where knowledge is captured in the form of rules to derive conclusions.

2. **Inference Rules:** These rules are used to derive new knowledge from existing knowledge. The most well-known types of inference rules are **Modus Ponens** and **Modus Tollens**.
    - **Modus Ponens**: If P implies Q, and P is true, then Q is true.
      Example:
      IF it rains THEN the ground will be wet.
      It is raining.
      Therefore, the ground is wet.
    - **Modus Tollens**: If P implies Q, and Q is false, then P is false.
      Example:
      IF it rains THEN the ground will be wet.
      The ground is not wet.
      Therefore, it is not raining.

3. **Frame-Based Rules:** Frames are data structures for representing stereotyped knowledge about objects and events. Rules can be embedded within frames to trigger actions when certain conditions about objects or events are met. These rules are usually used in **semantic networks** and **object-oriented knowledge bases**.

4. **Horn Clauses:** In logic programming, especially in languages like **Prolog**, knowledge is represented as Horn clauses. These clauses consist of a head and a body and are of the form:
   head :- body1, body2, ..., bodyN.
   This means that the head is true if all the conditions in the body are true.
   Example:
   rain(X) :- clouds(X), high_humidity(X).
   This rule means that it will rain if there are clouds and high humidity.

Components of Rule-Based Systems

1. **Knowledge Base:** This is where all the rules (knowledge) are stored. The knowledge base contains the facts and rules that are used by the system to draw inferences.
2. **Inference Engine:** The inference engine is the core of a rule-based system. It applies logical reasoning to the rules and facts in the knowledge base to deduce new facts or make decisions. There are two primary types of inference techniques:
    - **Forward Chaining**: The inference engine starts with the available facts and applies rules to infer new facts until a goal is reached. This is a data-driven approach. Example:

    - **Backward Chaining**: The inference engine starts with a goal and works backward to determine which facts must be true to achieve the goal. This is a goal-driven approach.
3. **Working Memory:** The working memory stores the facts that the system knows at any given time. It is updated as the inference engine applies rules to the facts in the knowledge base.
4. **Conflict Resolution:** When multiple rules can be applied at the same time, conflict resolution strategies are needed to decide which rule to fire first. Common strategies include:
    - **Priority-based resolution**: Assigning priorities to rules and firing the highest-priority rule.
    - **Specificity-based resolution**: Choosing the most specific rule to apply.
    - **Recency-based resolution**: Applying the rule that refers to the most recently updated facts.

Characteristics of Rule-Based Deduction Systems

1. **Deterministic:** Rule-based systems typically operate in a deterministic manner, meaning that given the same set of facts and rules, they will always produce the same result.
2. **Modular:** The rules in the knowledge base are independent of each other, allowing new rules to be added or modified without affecting the existing rules significantly.
3. **Explicit Knowledge Representation:** Knowledge in the system is represented explicitly through rules and facts, making it easier to understand and explain the reasoning process.
4. **Domain-Specific:** Rule-based systems are often designed for specific problem domains, such as medical diagnosis, financial decision-making, or legal reasoning. The effectiveness of the system depends on the quality and completeness of the rules.

Advantages of Rule-Based Knowledge Representation

1. **Modularity:** Rules can be added, modified, or removed independently, making the knowledge base easy to maintain and update.
2. **Transparency:** Rules are easy to understand, and the reasoning process can be explained in a way that humans can follow.
3. **Declarative Nature:** Rules focus on "what" needs to be done rather than "how" it should be done, allowing the system to represent knowledge at a high level of abstraction.
4. **Flexibility:** Rule-based systems can easily accommodate additional knowledge, allowing them to grow in complexity.
5. **Inference Mechanism:** The separation of the inference engine from the knowledge base allows rules to be applied in a variety of situations, making the system more adaptable.

Disadvantages of Rule-Based Knowledge Representation

1. **Scalability Issues:** As the number of rules grows, the system can become slower, and managing a large number of rules can become difficult.
2. **Inability to Learn:** Traditional rule-based systems cannot learn from new data. They are static and require manual updating of rules.
3. **Handling Uncertainty:** Rule-based systems can have difficulty dealing with uncertainty and incomplete information. Extensions, such as fuzzy logic, may be needed to handle these cases.
4. **Rule Conflicts:** In cases where two or more rules contradict each other, the system may encounter problems in deciding which rule to apply, requiring sophisticated conflict resolution mechanisms.

Applications of Rule-Based Systems

1. **Expert Systems:** Expert systems use rule-based knowledge representation to solve complex problems in domains like medical diagnosis, legal reasoning, and financial analysis. Examples include **MYCIN** for medical diagnosis and **DENDRAL** for chemical analysis.
2. **Decision Support Systems:** Rule-based systems are used in decision support systems to assist in decision-making processes. For instance, they can be used in credit scoring, risk assessment, and business rule management.
3. **Natural Language Processing:** Rule-based systems can be applied in natural language processing (NLP) for parsing sentences, performing semantic analysis, and understanding language patterns.
4. **Robotics and Automation:** In robotics, rules can control the actions and decision-making processes of autonomous systems based on sensory input and environmental conditions.

Example of a Rule-Based Deduction System

Consider a **loan approval system** that uses a set of rules to decide whether to approve or reject a loan application. The rules might include:

- **Rule 1**: IF (credit score > 700) AND (income > $50,000) THEN approve loan.
- **Rule 2**: IF (credit score <= 700) AND (income <= $50,000) THEN reject loan.

Given a set of facts such as:

- Applicant credit score: 750
- Applicant income: $55,000

The system will apply the rules and deduce that the loan should be approved based on Rule 1.

Procedure Vs Declarative Knowledge

Procedural Knowledge

Procedural knowledge is the knowledge of **how to perform tasks or processes**. It focuses on the **steps or procedures** needed to achieve a specific goal or solve a problem. This type of knowledge is typically action-oriented and involves a sequence of actions or rules to follow.

Key Characteristics:

- **"How" Knowledge**: Procedural knowledge explains how to do something or how to perform specific tasks.
- **Implicit**: Often, the rules are implicit and might not be easy to express in words, but are embedded in the process of performing a task.
- **Difficult to Verbalize**: Procedural knowledge is often hard to articulate but is demonstrated through action (e.g., riding a bicycle, solving an equation).
- **Executable Knowledge**: It provides instructions on how to carry out specific tasks or operations. For example, a set of rules to solve a Rubik's Cube is procedural knowledge.
- **Dynamic**: It requires execution and performance, as it's often applied in real-time to solve problems.

Examples of Procedural Knowledge:

- **Cooking a recipe**: Following step-by-step instructions to prepare a dish.
- **Solving a math problem**: Knowing the algorithm to solve a quadratic equation.
- **Driving a car**: Using the knowledge of how to coordinate steering, acceleration, and braking.
- **Programming a function**: Writing a function that sorts a list of numbers.

How It Is Represented in AI:

- In AI, procedural knowledge is often represented in the form of **algorithms** or **procedures** (e.g., functions, rules, control flows in programs).

- **Production systems** in AI (like rule-based systems) use procedural knowledge to perform operations when certain conditions are met.
- Example in AI: A search algorithm like **Depth-First Search (DFS)** or **Breadth-First Search (BFS)** is a procedural method for exploring nodes in a graph.

Advantages:

- Effective for **problem-solving** tasks where a sequence of operations or actions is necessary.
- Can be implemented directly in computational systems through algorithms and programming.

Limitations:

- **Hard to transfer**: Procedural knowledge is often specific to a particular context and might be hard to generalize.
- **Difficult to explain**: People with procedural knowledge may not always be able to explain the process easily to others, as much of it can be unconscious or intuitive.

Declarative Knowledge

Declarative knowledge is the knowledge of **facts and information** about the world. It represents **what is true** or describes the relationships between entities or concepts without specifying how to use that information in specific actions. It is static knowledge that can be easily articulated and shared.

Key Characteristics:

- **"What" Knowledge**: Declarative knowledge explains what things are, describes concepts, facts, or relationships, and is focused on providing information rather than processes.
- **Explicit**: Declarative knowledge is usually explicit and easy to verbalize or write down.
- **Static**: It represents facts and data rather than processes or sequences of operations.
- **Easily Shared**: Since declarative knowledge consists of facts and information, it is easier to communicate and transfer from one person to another.

Examples of Declarative Knowledge:

- **Historical facts**: Knowing that the moon landing happened in 1969.
- **Scientific knowledge**: Understanding the structure of an atom or the law of gravity.

- **Definitions**: Knowing the definition of a word or a mathematical theorem.
- **General knowledge**: Knowing that Paris is the capital of France.

How It Is Represented in AI:

- In AI, declarative knowledge is often represented in **knowledge bases**, **semantic networks**, **frames**, **ontologies**, or **logic-based representations** such as **Prolog** facts.
- **Rule-based systems**: In expert systems, declarative knowledge can be represented in the form of facts or rules that describe the knowledge about a domain.
    - Example: In a medical expert system, a declarative fact could be: "If a patient has a fever, they are likely to have an infection."

Advantages:

- **Easily transferable**: Since declarative knowledge is usually factual, it can be easily shared, taught, or modified.
- **Useful for reasoning**: Declarative knowledge forms the basis for reasoning and making logical deductions in AI systems.
- **Readable by humans**: Declarative knowledge is often human-readable and can be used to explain system decisions.

Limitations:

- **Doesn't explain processes**: Declarative knowledge doesn't include instructions on how to perform tasks. It lacks the procedural aspect needed for problem-solving.
- **May require interpretation**: To use declarative knowledge for practical tasks, an AI system often needs an inference engine to deduce how to apply the facts in a specific situation.

Differences between Procedural and Declarative Knowledge

Aspect	Procedural Knowledge	Declarative Knowledge
Definition	Knowledge of **how** to perform tasks or processes	Knowledge of **what** facts or information is true
Focus	Focuses on the **execution of tasks** and actions	Focuses on **facts, data, and relationships**
Nature	**Action-oriented** and dynamic; involves steps	**Fact-oriented** and static; represents information
Example	Knowing how to solve a math problem	Knowing that Paris is the capital of France
Representation in AI	Represented in algorithms, production rules, and functions	Represented in knowledge bases, ontologies, and logic
Execution Requirement	Requires execution (e.g., following a procedure)	Does not require execution, used for reasoning
Complexity	More complex to explain and transfer	Easier to explain and share
Usefulness	Useful for **problem-solving** and performing tasks	Useful for **reasoning, storing facts**, and understanding
Transferability	Hard to articulate and share	Easier to verbalize and communicate
Cognitive Load	Often intuitive or unconscious knowledge (implicit)	Explicit knowledge that can be easily stored and retrieved
Learning Type	Acquired through practice and experience	Learned through study, observation, or instruction
Use in AI	Used in search algorithms, control flows, and decision-making processes	Used in knowledge representation, expert systems, and inference engines
Performance	Directly affects system performance (how tasks are executed)	Enhances system's knowledge base for decision making