Lesson 4: Knowledge-Based Systems

4.1 Introduction

A knowledge-based system (KBS) is a form of artificial intelligence (AI) systems that aims to capture the knowledge of human experts to support decision-making and solve complex problems.

Examples of knowledge-based systems include expert systems, which are so called because of their reliance on human expertise.

4.2 Knowledge-based agents (KBA)

A knowledge base agent is an AI system that maintains a structured repository of information, often in the form of a database or knowledge graph. This knowledge is used to answer questions, make decisions, or perform other tasks.

4.2.1 Components of KBA

1. **Knowledge Base**: This is where information is stored. It can be in the form of structured data, facts, rules, or a combination of these. Knowledge bases are designed to represent knowledge in a way that the agent can reason over.
2. **Inference Engine**: The inference engine is responsible for making deductions and inferences based on the information stored in the knowledge base. It uses rules, logic, or algorithms to draw conclusions.
3. **User Interface**: Knowledge base agents typically have a user interface that allows users to interact with the system, input questions or queries, and receive responses.

4.2.2 Use Cases of KBA

1. **Question Answering**: Knowledge base agents can answer factual questions by retrieving information from their knowledge base. For example, they can provide information about historical events, scientific facts, or company data.
2. **Expert Systems:** These are knowledge base agents that mimic the decision-making abilities of a human expert in a specific domain. They are often used in fields like medicine and finance for diagnosis and advisory purposes.
3. **Personal Assistants:** Virtual personal assistants, like Siri and Google Assistant, use knowledge base agents to provide information, set reminders, and assist with various tasks.

4.3 KBA and logical problem-solving

Knowledge base agents (KBAs) are designed to handle logical problem-solving tasks by leveraging the information stored in their knowledge bases. Logical problem-solving involves making inferences and deductions based on available information to arrive at a solution. Here's how KBAs are utilized in logical problem-solving scenarios:

1. **Knowledge Representation**: KBAs store information in a structured format within their knowledge bases. This structured representation might include facts, rules, ontologies, frames, or semantic networks. These representations allow the KBA to understand the relationships between different pieces of information.
2. **Rule-Based Reasoning**: KBAs often use a set of rules or logical axioms to perform reasoning. When a user presents a problem or query, the KBA's inference engine applies these rules to the

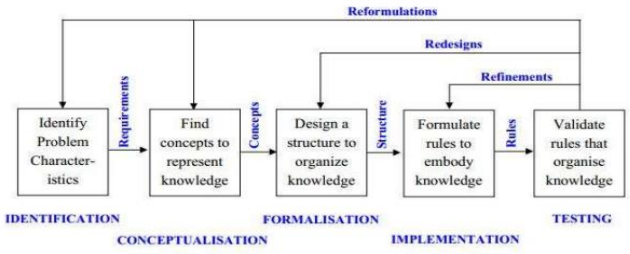
known facts in the knowledge base. Through this rule-based reasoning, the agent can draw logical conclusions.

3. **Deductive Reasoning**: Deductive reasoning involves reaching specific conclusions based on general principles. KBAs use deductive reasoning to infer new facts or relationships from the existing knowledge in the knowledge base. For example, if the KBA knows that "all humans are mortal" and "Socrates is a human," it can deduce that "Socrates is mortal."
4. **Abductive Reasoning:** Abductive reasoning involves making educated guesses or hypotheses to explain observed phenomena. KBAs can use abductive reasoning to generate possible explanations for a given problem. The KBA generates hypotheses based on available information and then tests these hypotheses against additional knowledge to see if they are plausible explanations.
5. **Problem Solving with Heuristics:** KBAs can incorporate heuristics or problem-solving strategies to efficiently search through the solution space. By using heuristics, KBAs can prioritize certain paths or solutions, making the problem-solving process more efficient.
6. **Constraint Satisfaction Problems**: KBAs can handle problems where the solution must satisfy a set of constraints. For example, scheduling problems, timetabling, and configuration tasks often involve constraints. KBAs can use constraint satisfaction techniques to find solutions that meet all the specified conditions.
7. **Query Expansion and Reformulation**: When faced with vague or ambiguous queries, KBAs can expand or reformulate the query based on the context and available knowledge. This process helps in narrowing down the search space and finding relevant information to address the problem.
8. **Learning and Adaptation**: Advanced KBAs can incorporate machine learning techniques to adapt and improve their problem-solving abilities over time. By learning from past interactions and feedback, KBAs can enhance their knowledge and reasoning capabilities.

4.4 Knowledge acquisition

Knowledge acquisition is the process of extracting, structuring and organizing knowledge from one source, usually human experts, so it can be used in software such as an Expert Systems (ES). It can also be defined as the process of extracting knowledge (facts, procedures, rules) from human experts, books, documents, sensors or computer files and converting it into a form that can be stored and manipulated by the computer for purposes of problem solving. It occurs throughout the entire development process.

The iterative nature of the knowledge acquisition process can be represented diagrammatically as below



4.5 Knowledge representation

Knowledge representation (KR) is the field of artificial intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language. Knowledge representation incorporates findings from psychology about how humans solve problems and represent knowledge in order to design formalisms that will make complex systems easier to design and build. Knowledge representation and reasoning also incorporates findings from logic to automate various kinds of reasoning, such as the application of rules or the relations of sets and subsets There is always a relationship between the form in which knowledge is represented and the way in which the knowledge is used. You can use domain specific or general-purpose representation.

For a knowledge-based intelligent program, we need:

i. To represent knowledge about the world in a formal language.
ii. To reason about the world using inferences in the language.
iii. To decide what action to take by inferring that the selected action is good.

KR languages – are special notations that allows to easily represent complex facts:

a) Adequately.
b) Clearly and precisely.
c) Naturally.

Various ways for KR include:

a) **Semantic Networks-** a graph theoretic data structure whose nodes represent word senses and whose arcs express semantic relationships between these word senses.
b) **Scripts**- a structure that describe a sequence of events in particular context scripts are frame like structures used to represent commonly occurring experiences such as going to movies, shopping in supermarket, eating in restaurant, banking.
c) **Rules**- the knowledge structure that relates some known information to other information that can be concluded or inferred from the former.
d) **Frames**- a record like structure which consists of a collection of attributes and its values to describe an entity in the world.

4.6 Propositional logic

Propositional logic, also known as sentential logic, is a fundamental branch of formal logic that deals with propositions, which are statements or declarative sentences that can be either true or false, but not both. It provides a systematic and mathematical way of reasoning about the truth or falsity of statements and how they relate to one another. It serves as the foundation for more complex logical systems and formal methods for knowledge representation and problem-solving

Here are the key components and concepts of propositional logic:

1. **Propositions (Statements):** In propositional logic, propositions are the basic units of information. They are represented by variables (usually p, q, r, etc.) and can be assigned truth values (True or False). For example, "It is raining" can be represented by the variable p.
2. **Connectives (Logical Operators):** Connectives are used to combine or manipulate propositions. The primary connectives in propositional logic include:
    a. **Conjunction (∧):** Represents "and." The conjunction of two propositions is true only if both propositions are true.
    b. **Disjunction (∨):** Represents "or." The disjunction of two propositions is true if at least one of the propositions is true.
    c. **Negation (¬):** Represents "not." Negation flips the truth value of a proposition.
    d. **Implication (→):** Represents "if...then..." An implication is true unless the antecedent (the part before the arrow) is true and the consequent (the part after the arrow) is false.
    e. **Biconditional (↔):** Represents "if and only if." A biconditional is true if both sides have the same truth value.
3. **Truth Tables:** Truth tables are used to systematically evaluate the truth value of compound propositions by considering all possible combinations of truth values for their constituent propositions. Truth tables provide a way to determine the truth value of complex logical expressions.
4. **Logical Equivalences:** These are rules and relationships that allow you to simplify or transform logical expressions while preserving their truth values. For example, De Morgan's laws state that ¬(p ∧ q) is equivalent to (¬p ∨ ¬q) and ¬(p ∨ q) is equivalent to (¬p ∧ ¬q).
5. **Tautologies and Contradictions**: A tautology is a compound proposition that is always true, regardless of the truth values of its constituent propositions. A contradiction is a proposition that is always false. For example, (p ∨ ¬p) is a tautology, and (p ∧ ¬p) is a contradiction.
6. **Inference Rules**: In propositional logic, you can use inference rules such as Modus Ponens and Modus Tollens to make deductions based on premises. For example, Modus Ponens states that if you have (p → q) and p, you can conclude q.

4.7 First-order logic (FOL)

FOL is a more expressive and complex formal logical system than propositional logic. It is also known as first-order predicate calculus. First-order logic extends propositional logic by introducing variables, quantifiers, and predicates, making it suitable for more intricate and precise reasoning about the real world. Here are the fundamental components and concepts of first-order logic:

1. **Variables**: In first-order logic, variables are used to represent objects or elements in the domain of discourse. Variables are typically denoted by lowercase letters (x, y, z, etc.).
2. **Constants**: Constants are specific objects in the domain of discourse. They are used to represent particular individuals or objects. Constants are often denoted by lowercase letters or specific symbols (e.g., a, b, c).
3. **Predicates**: Predicates are used to express relationships or properties that can hold between objects. Predicates are denoted by uppercase letters or symbols (e.g., P, Q) and can take one or more arguments. For example, P(x) might represent "x is blue," where x is a variable.

4. **Quantifiers**: First-order logic includes two quantifiers:
    a. **Universal Quantifier (∀):** Represents "for all" or "for every." It is used to express statements that hold true for all elements in the domain of discourse. For example, ∀x P(x) means "For all x, x is blue."
    b. **Existential Quantifier (∃):** Represents "there exists." It is used to express statements that hold true for at least one element in the domain of discourse. For example, ∃x P(x) means "There exists an x such that x is blue."
5. **Logical Connectives:** First-order logic includes the same logical connectives as propositional logic, such as conjunction (∧), disjunction (∨), negation (¬), implication (→), and biconditional (↔).
6. **Functions:** Functions are used to represent operations that map objects from the domain of discourse to other objects. Functions are denoted by lowercase letters and can take one or more arguments. For example, f(x) might represent "the father of x," where x is an individual.
7. **Terms:** Terms in first-order logic are expressions formed by combining constants, variables, and functions. Terms are used to refer to specific individuals or objects in the domain.
8. **Formulas:** Formulas are statements in first-order logic that are built using predicates, quantifiers, logical connectives, and terms. Formulas can be either atomic (simple, with no logical connectives) or compound (containing logical connectives).
9. **Axioms and Inference Rules:** First-order logic employs axioms and inference rules to derive conclusions from a set of premises. Common inference rules include Modus Ponens, Universal Generalization, Existential Instantiation, and more.
10. **Models and Interpretations:** In first-order logic, a model is an interpretation of the logical symbols within a specific domain of discourse. A model assigns meaning to the predicates, functions, and constants, making it possible to evaluate the truth or falsity of first-order logic statements within that model.

Lesson 4: Review Questions

1. Discuss top-down and bottom-up knowledge acquisition strategies.
2. Differentiate Predicate Logic vs. Propositional Logic