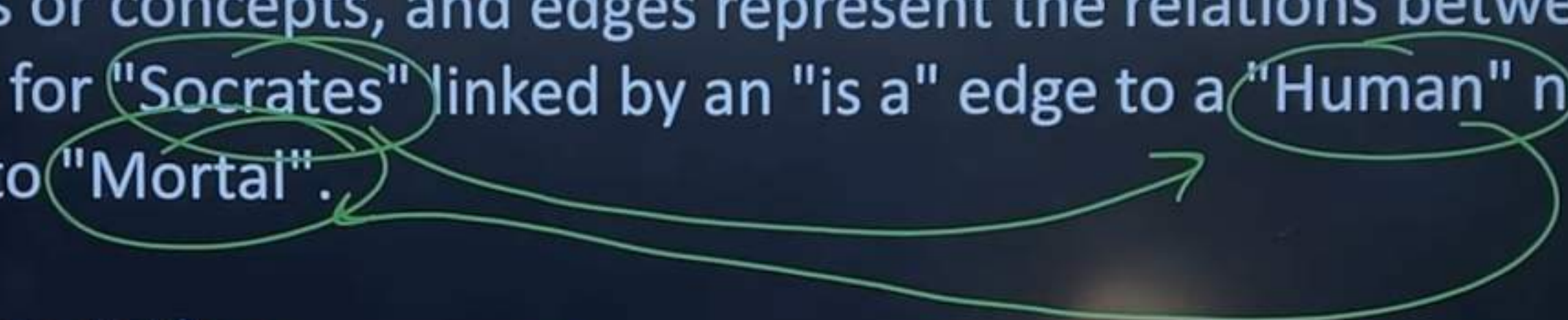
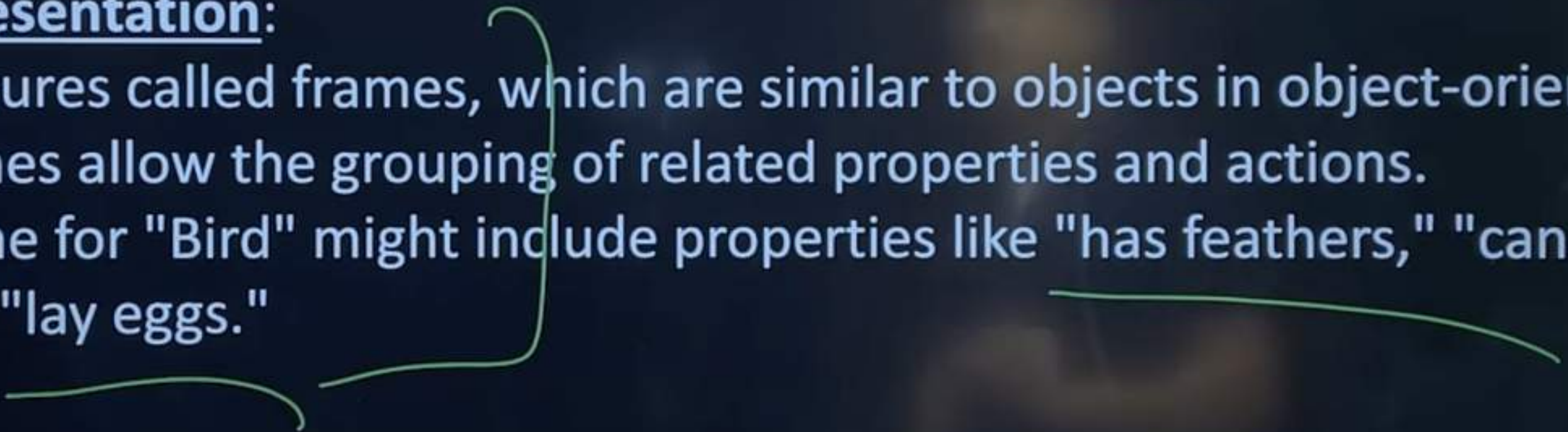


### Semantic Networks:

- Graph structures used to represent semantic relations between concepts. Nodes represent objects or concepts, and edges represent the relations between them.
  - Example: A node for "Socrates" linked by an "is a" edge to a "Human" node, and "Human" linked to "Mortal".
- 

### Frame-Based Representation:

- Uses data structures called frames, which are similar to objects in object-oriented languages. Frames allow the grouping of related properties and actions.
  - Example: A frame for "Bird" might include properties like "has feathers," "can fly," and actions like "lay eggs."
- 



### • Production Rules:

- Consist of sets of rules in the form of IF-THEN constructs that are used to derive conclusions from given data.
- Example: IF the patient has a fever and rash, THEN consider a diagnosis of measles.

### • Ontologies:

- Formal representations of a set of concepts within a domain and the relationships between those concepts. They are used to reason about the entities within that domain and are often employed in semantic web applications.
- Example: In a medical ontology, concepts like "symptom," "disease," and "treatment" might be related in ways that define what symptoms are commonly associated with a disease.



### • Bayesian Networks:

- Probabilistic models that represent a set of variables and their conditional dependencies via a directed acyclic graph (DAG). These are useful for handling uncertainty in AI applications.
- Example: A network might represent the probabilistic relationships between diseases and symptoms, allowing the system to infer disease probabilities given observed symptoms.

### • Neural Networks:

- Although typically classified under machine learning, neural networks can represent complex relationships between inputs and outputs through learned weights and can effectively capture and model knowledge after training.
- Example: A neural network trained on historical weather data might predict future weather conditions.



## Chapter-3 (KNOWLEDGE REPRESENTATIONS)

### Knowledge Representation

- **Definition**: It's the study of how knowledge can be represented in a computer system to mimic human thought and reasoning.
- **Components**:
  - Facts: True information relevant to a specific domain.
  - Representation: How facts are formally expressed, often using logical structures.
- **Example**: For the fact "Charlie is a dog," a logical representation could be Dog(Charlie).
- **Characteristics**:
  - **Syntax and semantics**: Clearly defined rules for structure and meaning.
  - **Expressive capacity**: Ability to convey all necessary domain knowledge.
  - **Efficiency**: Supports effective computing and reasoning.



### Fact:

"Book X is available in the library."

### Representation in Logical Structure:

Using a logical statement, this fact can be represented as:

`Available(Book X)`

### Characteristics Applied:

- **Syntax and Semantics:** The function `Available()` clearly defines that it checks for the availability of a specific book.
- **Expressive Capacity:** This structure allows the system to represent the availability of any book in the library by substituting "Book X" with any book's name.
- **Efficiency:** Such representation supports quick queries about book availability, facilitating efficient information retrieval and decision-making processes in the library's computer system.



- **Definition:**

- Involves gathering expertise for use in expert systems, organizing it into structured formats like IF-THEN rules.
- Also refers to the process of absorbing information into memory, focusing on effective retrieval.

- **Procedure:**

- **Identification:** Break down complex issues into parts.
- **Conceptualization:** Define key concepts.
- **Formalization:** Organize knowledge formally for programmatic use.
- **Implementation:** Code the structured knowledge.
- **Testing:** Check and validate the system's functionality.



To create an expert system that helps diagnose skin diseases based on history.

## Chapter-3 (KNOWLEDGE REPRESENTATIONS)

### Knowledge Acquisition Process:

#### 1. Identification:

- Break down the domain of skin diseases into manageable categories, such as infectious, non-infectious, inflammatory, and allergic skin conditions.

#### 2. Conceptualization:

- Define key concepts and terms, such as "eczema," "psoriasis," and "dermatitis." Understand the common symptoms associated with each category, like redness, itching, or peeling.

#### 3. Formalization:

- Organize the knowledge into a structured format:
  - IF the patient exhibits symptoms X, Y, and Z, AND has a history of A,
  - THEN the likely diagnosis is B.
- Example rule: IF the patient has red, itchy patches, AND has a family history of allergies, THEN consider "eczema" as a diagnosis.

#### 4. Implementation:

- Code the knowledge into the expert system using a suitable programming language, integrating decision-making logic based on the IF-THEN rules.

#### 5. Testing:

- Validate the system's functionality by inputting test cases (e.g., symptoms of known diseases) to see if the expert system correctly diagnoses them based on the programmed knowledge.

[WWW.KNOWLEDGEGATE.IN](http://WWW.KNOWLEDGEGATE.IN)



In artificial intelligence (AI), knowledge representation is a critical area that enables systems to simulate human-like reasoning and decision-making. Various techniques are employed to represent knowledge in AI systems, each with specific applications and advantages. Here are some key techniques:

### Logic-Based Representation:

- Propositional Logic: Uses simple statements that are either true or false.
- Predicate Logic: Extends propositional logic with predicates that can express relations among objects and quantifiers to handle multiple entities.
- Example: Representing the relationship, "All humans are mortal," can be written in predicate logic as  $\forall x (\text{Human}(x) \rightarrow \text{Mortal}(x))$ .



Existential Instantiation

- Existential instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic. It can be applied only once to replace the existential sentence.
- This rule states that one can infer  $P(c)$  from the formula given in the form of  $\exists x P(x)$  for a new constant symbol  $c$ . The restriction with this rule is that  $c$  used in the rule must be a new term for which  $P(c)$  is true.

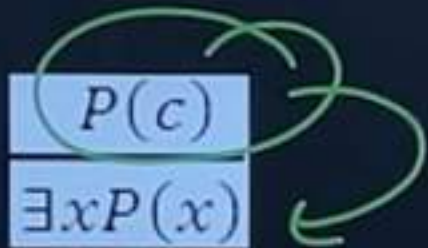
$$x = \{1, 2, 3, \dots\}$$

- It can be represented as:

$$\frac{\exists x P(x)}{P(c)}$$



## Existential introduction

- An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic. This rule states that if there is some element  $c$  in the universe of discourse which has a property  $P$ , then we can infer that there exists someone in the universe which has the property  $P$ .
- It can be represented as: The diagram shows a box containing  $P(c)$  above a box containing  $\exists xP(x)$ . A green arrow points from the  $P(c)$  box to the  $\exists xP(x)$  box. A green circle is drawn around the  $P(c)$  box, and a green arrow points from this circle to the  $\exists$  quantifier in the  $\exists xP(x)$  box.
- **Example:** "Priyanka got good marks in English."
- "Therefore, someone got good marks in English."



Universal Generalization

- Universal generalization is a valid inference rule which states that if premise  $P(c)$  is true for any arbitrary element  $c$  in the universe of discourse, then we can have a conclusion as  $\forall x P(x)$ .
- It can be represented as: 
$$\frac{P(c)}{\forall x P(x)}$$
- This rule can be used if we want to show that every element has a similar property. In this rule,  $x$  must not appear as a free variable.
- **Example:** Let's represent,  $P(c)$ : "A byte contains 8 bits", so for  $\forall x P(x)$  "All bytes contain 8 bits.", it will also be true.

$$U = \{1, 2, 3, 4\}$$

$$P(1) \quad P(2) \quad P(3) \quad P(4)$$

$$\forall x P(x)$$



- Begins with what is known: starts with basic facts or data.
- Moves forward: applies rules to these facts to derive new information.
- Data-driven: new conclusions are made as data is processed.
- Adds knowledge: builds upon existing information incrementally.
- Stops when no further deductions can be made or a goal is reached.



### Example: A Rice Crop Management System

- **Initial Observation:** A farmer in Odisha notices that some rice plants are shorter than usual and have yellow leaves.
- **Rule 1:** If rice plants are short and have yellow leaves, they might lack nitrogen.
- **Rule 2:** If rice plants lack nitrogen, then using nitrogen-rich fertilizer may help.
- **Rule 3:** If the season is monsoon and the problem is nitrogen deficiency, then use urea fertilizer, which is suitable for wet conditions.

### Forward Chaining Process:

1. **Observing Symptoms:** The farmer inputs into the system that the rice plants are short with yellow leaves.
2. **Applying Knowledge:** The system uses Rule 1 to deduce that the plants might be suffering from nitrogen deficiency.
3. **Taking Action:** Based on Rule 2, the system suggests that applying a nitrogen-rich fertilizer could be beneficial.
4. **Considering Conditions:** Seeing that it is the monsoon season, Rule 3 advises the farmer to use urea fertilizer specifically.
5. **Outcome:** The farmer follows the advice, applies urea, and over time the rice plants become healthier and grow taller.



- Starts with a goal: looks at the desired outcome first. ✓
- Works backward: checks if current knowledge supports the goal.
- Goal-driven: each step is aimed at proving the end objective.
- Deduces necessities: determines what conditions must be met for the goal.
- Stops when it reaches the beginning facts or rules, or the goal is proven or disproven.



Example: Deciding on a Postgraduate Course

- **Goal:** A student wants to determine the best postgraduate course to enroll in.
- **Question:** What course should the student choose for a successful career?
- **Check Desired Outcome:** The desired outcome is a fulfilling and prosperous career in a field the student is passionate about.
- **Compare Interests:** The student is interested in technology and innovation.
- **Search for Career Paths:** The system considers various career paths that align with technology and innovation.
- **Hypothesis 1:** Is a Master's in Computer Science suitable?
- **Check for Evidence:** The system checks the student's academic background, skills, and job market trends for computer science graduates.
- **Hypothesis 2:** Could an MBA in Technology Management be an alternative?
- **Check for Evidence:** The system evaluates the student's leadership potential, market demand for technology managers, and the student's long-term career goals.
- **Solution if Matched:** If the evidence strongly supports one path over the other based on the student's profile and job market trends, the system recommends that course.
- **Outcome:** The student chooses the recommended course which aligns with their interests, background, and market demand, leading to a more focused educational path towards a desired career.



## Forward Chaining

## Backward Chaining

1. Data-driven approach: begins with known facts.

1. Goal-driven approach: starts with the desired conclusion.

2. Bottom-up reasoning: builds up from facts to a conclusion.

2. Top-down reasoning: works backwards from goal to facts.

3. Uses Modus Ponens: applies rules to derive new information.

$$\begin{array}{l} P \rightarrow Q \\ P \\ \hline Q \end{array}$$

3. Uses Modus Tollens: checks if conclusions can lead to the goal.

$$\begin{array}{l} P \rightarrow Q \\ \sim Q \\ \hline \sim P \end{array}$$

4. Continues until no new information can be derived.

4. Continues until it reaches the starting facts or rules.

5. Common in real-time systems and production rule systems.

5. Used in diagnostic systems where the end goal is known.



- Resolution in AI is a logical inference rule used to deduce new information or solve problems by finding contradictions.
- It involves converting logical statements into a standardized form called clausal form, which is a set of normalized logical expressions.
- Contradictory pairs of clauses are identified, and through a process of unification and combination, new clauses are generated.
- This process is repeated iteratively until either a contradiction is found (represented by an empty clause) or no new information can be deduced.
- Resolution is particularly powerful in automated theorem proving and logic programming, such as in Prolog, where it's used to infer conclusions from given facts and rules.



- All birds fly:  $\forall x (\text{Bird}(x) \rightarrow \text{Fly}(x))$
- Tweety is a bird:  $\text{Bird}(\text{Tweety})$

We want to prove that Tweety can fly. Using resolution:

1. We negate the query  $\neg \text{Fly}(\text{Tweety})$  and add it to our knowledge base.

2. Our clauses in clausal form are:

- $\neg \text{Bird}(x) \vee \text{Fly}(x)$  (from  $\forall x (\text{Bird}(x) \rightarrow \text{Fly}(x))$ )
- $\text{Bird}(\text{Tweety})$
- $\neg \text{Fly}(\text{Tweety})$  (negated query)

$$P \rightarrow Q \equiv \neg P \vee Q$$

3. Now we apply resolution:

- We resolve  $\neg \text{Fly}(\text{Tweety})$  with  $\neg \text{Bird}(x) \vee \text{Fly}(x)$  using unification by substituting  $x$  with Tweety.
- This gives us  $\neg \text{Bird}(\text{Tweety}) \vee \text{Fly}(\text{Tweety})$ .

4. Finally, we resolve  $\text{Bird}(\text{Tweety})$  with  $\neg \text{Bird}(\text{Tweety}) \vee \text{Fly}(\text{Tweety})$ , which gives us  $\text{Fly}(\text{Tweety})$ , the desired fact.

Since we started with  $\neg \text{Fly}(\text{Tweety})$  and ended up with  $\text{Fly}(\text{Tweety})$ , we have derived a contradiction, meaning our original negation of the query must be false, thus proving that, according to our knowledge base, Tweety can fly.

WWW.KNOWLEDGEGATE.IN