Study Material

Table of Contents

Module No.	Module Name	Content
Module 2	Knowledge Representation Scheme and Reasoning Concept	First Order Logic

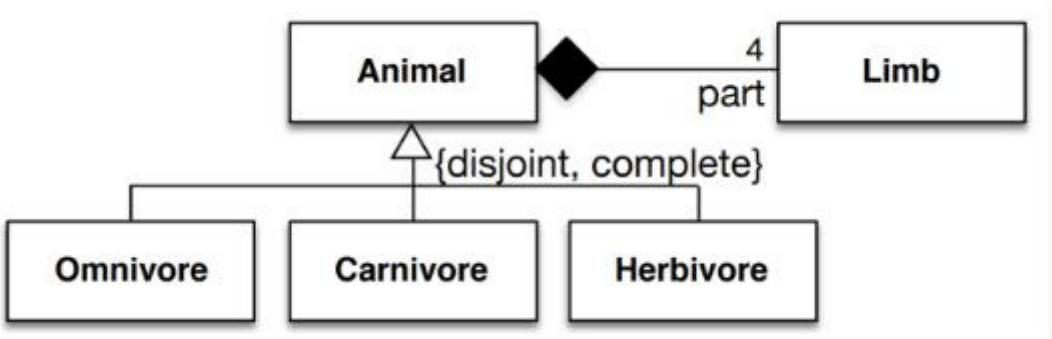
- **Importance of Knowledge Representation and Reasoning**
  - To build intelligent systems, it is crucial to have a **formal, explicit representation of the world and its workings**.
  - **Knowledge representation** and **automated reasoning** are essential capabilities for an AI system, enabling agents to answer questions, draw new conclusions, and make good decisions.
  - Historically, the **logicist tradition** within AI hoped to build intelligent systems based on programs that could solve problems described in logical notation.

I. Introduction

- **What is Logic?**
  - Logic is fundamental in both cognitive science and artificial intelligence (AI).
  - In cognitive science, it addresses how people store and process information.
  - In AI, the main focus is on storing knowledge so that programs can process it and achieve human-like intelligence.
  - First-Order Logic (FOL), also known as "quantificational logic" or "predicate logic," is a formal language designed for precise reasoning.
  - It provides a formal analogue of English sentences, allowing logicians to ask questions about logical consequences, truth, falsity, and equivalence.

I. First-Order Logic (FOL)

- **A. What is First-Order Logic?**
  - **Definition**: First-Order Logic (FOL) is a crucial concept in AI, especially in areas like theorem proving and logic programming.
  - It extends Boolean logic (developed by George Boole) to **include objects and relations**. Gottlob Frege extended Boole's logic to create the first-order logic used today.
  - In logic, **unification plays a critical role** in FOL, where it is used to **match formulas by finding common substitutions**.
  - **Key aspects of FOL**:
    - **Representation**: Allows for precise statements about objects in the world and relations among them.
    - **Inference**: Provides formal rules to draw valid conclusions.

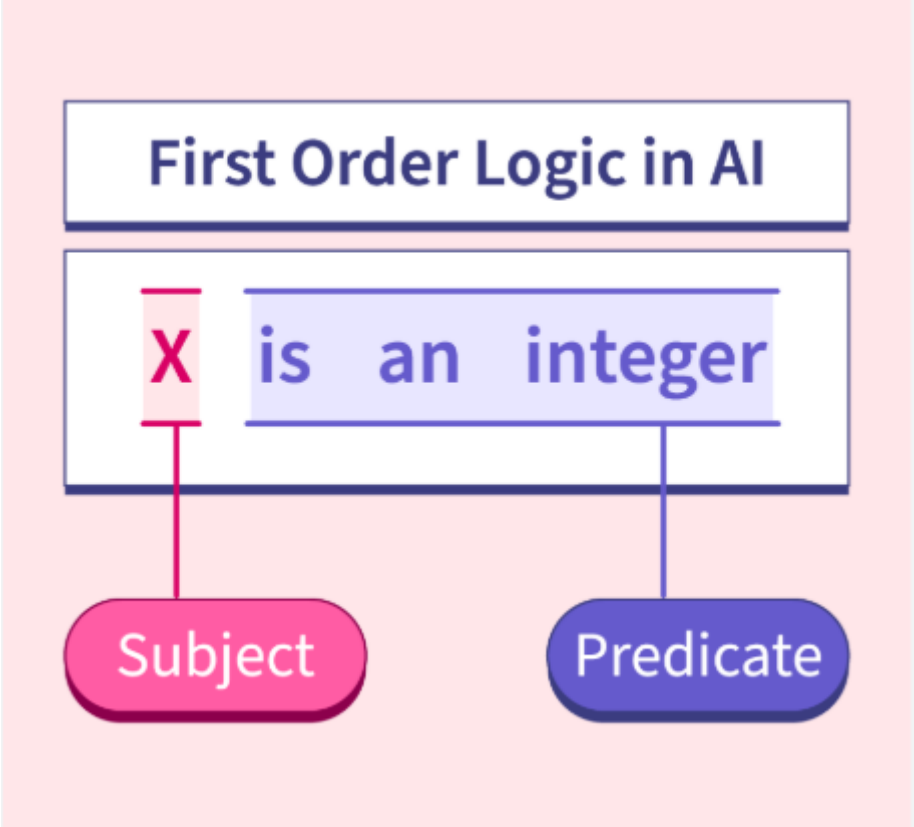


Structure of First Order Logic

- **B. Unification in First-Order Logic**
  - **Core Concept**: In FOL, unification is used to **determine if two predicates or terms can match by substituting variables**. This is fundamental for inference in AI domains like theorem proving and logic programming.
  - **Mathematical Perspective**: Unification involves finding a **substitution (θ)** for two or more expressions (E1 and E2) such that applying the substitution makes them equal: **E1θ = E2θ**.
    - **Substitution (θ)**: A mapping from variables to terms.

- **Most General Unifier (MGU)**: The simplest substitution that satisfies the unification.

- **Example of Unification in Terms**:
  - Consider two terms: $f(x, y)$ and $f(a, z)$.
  - We can unify these terms by finding the substitution $\theta = \{x/a, y/z\}$.
  - After applying θ, both terms become $f(a, z)$.
  - Thus, $\{x/a, y/z\}$ is the most general unifier (MGU) for these terms.
- **Unification Process in First-Order Logic (Atomic Formulas)**:
  - Given two atomic formulas, for example, $P(x, y)$ and $P(a, z)$.
  - The aim is to find a substitution θ that transforms both formulas into the same form.
  - Here, $\theta = \{x/a, y/z\}$ transforms both into $P(a, z)$.
  - This unification is what allows **reasoning systems to proceed with proofs or deductions**.
- **Unification Algorithm Steps in Logic Systems**:
  1. **Check for syntactic equality**: If both terms are identical, they are already unified.
  2. **Identify variables**: If one term contains variables, attempt to find a substitution that makes both terms equal.
  3. **Apply substitution**: Substitute the variables with terms to check if the expressions match.
  4. **Find the most general unifier (MGU)**: If multiple substitutions are possible, choose the simplest one.
- **Significance**: Unification ensures that **AI systems based on FOL can efficiently reason**.



Structure of Sentence in First Order Logic

First order logic – Contains predicates, quantifiers and variables

E.g. Philosopher(a) -> Scholar(a) • ∀x, King(x) ∧ Greedy (x) ⇒ Evil (x)

Squares neighboring the wumpus are smelly

– Objects: Wumpus, squares

– Property: Smelly

– Relation: neighboring

• Evil king john rules England in 1200

– Objects: John, England, 1200

– Property: evil, king

– Relation: ruled

- **C. Inference in First-Order Logic**
  - Chapter 9 of "Artificial Intelligence: A Modern Approach" specifically addresses **inference in First-Order Logic**.

- It contrasts **propositional vs. first-order inference**.
- Key inference mechanisms include:
  - **Forward Chaining**: A data-driven approach where inference moves from known facts to conclusions.
  - **Backward Chaining**: A goal-driven approach where inference starts from the goal and works backward to find facts that support it. Logic programming languages like Prolog rely heavily on backward chaining for inference.
  - **Resolution**: A complete **theorem-proving algorithm for first-order logic**. J.A. Robinson's discovery of the resolution method in 1965 advanced the plan to use logic to build intelligent systems.

Representing Facts in First-Order Logic

1. Lucy* is a professor
2. All professors are people.
3. John is the dean.
4. Deans are professors.
5. All professors consider the dean a friend or don’t know him.
6. Everyone is a friend of someone.
7. People only criticize people that are not their friends.
8. Lucy criticized John .

Same example, more formally

Knowledge base:

- is-prof(lucy)
- ∀ x (is-prof(x) → is-person(x))
- is-dean(John)
- ∀ x (is-dean(x) → is-prof(x))
- ∀ x (∀ y (is-prof(x) ∧ is-dean(y) → is-friend-of(y,x) ∨ ¬knows(x, y)))
- ∀ x (∃ y (is-friend-of (y, x)))
- ∀ x (∀ y (is-person(x) ∧ is-person(y) ∧ criticize (x,y) → ¬is-friend-of (y,x)))
- criticize(lucy, John)

Question: Is John no friend of Lucy?

¬is-friend-of(John ,lucy)

How the machine "sees" it:

Knowledge base:

- P1(A)
- ∀ x (P1(x) → P3(x))
- P4(B)
- ∀ x (P4(x) → P1(x))
- ∀ x (∀ y (P1(x) ∧ P4(y) → P2(y,x) ∨ ¬P5(x, y)))
- ∀ x (∃ y (P2(y, x)))
- ∀ x (∀ y (P3 (x) ∧ P3(y) ∧ P6(x,y) → ¬P2(y,x)))
- P6(A, B)

Question: ¬P2(B ,A)?

- **D. Knowledge Engineering in FOL**
  - FOL is used in **knowledge engineering** to represent information.
  - John McCarthy's "Programs with Common Sense" (1958) proposed the **Advice Taker**, a hypothetical program designed to embody general knowledge of the

world and use logic to derive plans of action. It was also designed to accept new axioms, allowing it to achieve competence in new areas without reprogramming.

- This concept embodied the central principles of **knowledge representation and reasoning**: using a formal, explicit representation of the world and manipulating it with deductive processes.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Rules of Knowledge Engineering

- **E. Limitations of Traditional FOL in AI**
  - Logic as conventionally understood requires **knowledge of the world that is certain**, a condition seldom achieved in reality.
  - The theory of probability helps to **fill this gap by allowing rigorous reasoning with uncertain information**.

- **Why Study First-Order Logic?**
    - FOL allows us to represent knowledge in a structured and unambiguous way, which is crucial for AI systems.
    - It forms the basis for many other advanced knowledge representation techniques and logical systems.
    - The methods developed for FOL are widely applied in logic, computer science, and linguistics.

II. Syntax of First-Order Logic: Building the Language

- **Syntax** defines what strings of symbols count as legitimate expressions in a formal language.
    - A formal language has a specific **vocabulary** and rules for forming expressions.
- **Basic Building Blocks (Vocabulary):** The vocabulary of FOL is divided into two parts.
    - **Logical Symbols:** These are fixed and provide the structure of logical statements.
        - **Logical Connectives:** Negation (¬), Conjunction (∧), Disjunction (∨), Conditional (→), Biconditional (↔).
        - **Quantifiers:** Universal (∀ for "for all"), Existential (∃ for "there exists").
        - **Identity Predicate:** The two-place identity predicate (=).
        - **Propositional Constants:** Falsity (⊥) and Truth (⊤).
        - **Variables:** A denumerable set of symbols like v0, v1, v2, ... (often x, y, z are used informally).
    - **Non-Logical Symbols (Language-Specific):** These give specific content to the logic and define a particular "first-order language" L.
        - **Predicate Symbols:** Names for properties or relations (e.g., P for "is prime," < for "is less than"). They have a specified arity (number of arguments).
        - **Constant Symbols:** Names for individual objects (e.g., a, b, c; or 0, 1 in arithmetic).
        - **Function Symbols:** Names for mappings or operations (e.g., f for "father of," + for "addition"). They also have a specified arity.
    - **Punctuation Marks:** Parentheses ( ) and commas (,).

Terms and Formulas

- **Terms:** Expressions that refer to objects in the domain. They are like nouns or noun phrases in natural language.
    - **Definition:**
        1. Every variable is a term (e.g., $v0$, x).
        2. Every constant symbol is a term (e.g., a, 0).
        3. If f is an n-place function symbol and $t1, ..., tn$ are terms, then $f(t1, ..., tn)$ is a term (e.g., $f(c0, v0)$, $(x + y)$).
    - A **closed term** is a term containing no variables (e.g., $f(c0, c1)$ or $(0 + 1)$ in arithmetic).
- **Formulas:** Expressions that state something true or false about objects or relations. They are like sentences in natural language.
    - **Atomic Formulas:** The simplest formulas.
        1. $\bot$ (Falsity) and $\top$ (Truth) are atomic formulas.
        2. If R is an n-place predicate symbol and $t1, ..., tn$ are terms, then $R(t1, ..., tn)$ is an atomic formula (e.g., $P(a)$, $Love(john, mary)$).
        3. If $t1$ and $t2$ are terms, then $=(t1, t2)$ (often written as $t1 = t2$) is an atomic formula.
    - **Complex Formulas:** Built from existing formulas using logical operators.
        1. If φ is a formula, then $\neg\varphi$ is a formula.
        2. If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are formulas. (Note: parentheses are crucial for unique readability).
        3. If φ is a formula and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.
- **Variables: Free vs. Bound:**
    - An occurrence of a variable x in a formula φ is **bound** if it falls within the scope of a quantifier $\forall x$ or $\exists x$.
    - An occurrence of a variable x is **free** if it is not bound.
    - **Scope:** The part of a formula to which a quantifier applies.
- **Sentences:** A formula φ is a **sentence** if it contains no free occurrences of variables.
    - Sentences represent complete propositions whose truth value can be determined without further context.
- **Unique Readability:** The careful inductive definition of formulas ensures that every formula has only one way of being constructed and interpreted, avoiding ambiguity.

- **Relevance to AI:**
    - First-order predicate logic is a common way to represent knowledge in AI systems.
    - Other knowledge representation (KR) approaches build upon or relate to FOL:
        - **Semantic Networks:** Graphical representation of objects, concepts, and relationships, some of which are formally defined systems of logic. Definitional networks (like description logics) are often subsets of FOL. Assertional networks assert propositions. Implicational networks use implication as the primary relation.
        - **Frames:** More structured packaging of knowledge, using hierarchies and inheritance, where "property links" from semantic nets are replaced by "SLOT fields".
        - **Conceptual Dependency Theory:** Developed by Schank to represent the meaning of natural language sentences using conceptual primitives, independent of language. Inferences are associated with primitive acts.
        - **Script Structure:** Larger structures building on conceptual dependency, representing stereotypical sequences of events (e.g., restaurant script).
        - **CYC Theory:** An ambitious attempt to build a very large knowledge base for commonsense reasoning, encoding implicit and explicit knowledge using a LISP-like, frame-based language.
        - **Case Grammars:** Linguistic analysis focusing on deep semantic roles (Agent, Object, Benefactor, Location, Instrument) required by verbs, forming "case frames".