Chapter VIII - First-Order Logic Inference

8.1 Propositional Inference vs First-Order Inference

Inference is the process of deriving new information or conclusions from known facts or premises. The primary distinction between **propositional inference** and **first-order inference** lies in the complexity and richness of the logic used.

1. **Propositional Inference:**
   - Propositional logic operates with simple, atomic propositions (e.g., "P", "Q", "R").
   - Inference involves logical rules (like **Modus Ponens**) that derive conclusions based on the truth values of these atomic propositions.
   - Inference in propositional logic is limited to statements without relationships, quantifiers, or variables.
2. **First-Order Inference:**
   - First-Order Logic (FOL) can express relationships between objects, properties, and functions, allowing more complex reasoning.
   - FOL includes **quantifiers** (e.g., `∀x`, `∃x`) and **predicates** to reason about objects and their relationships.
   - Inference in FOL is much more powerful as it can handle universally quantified statements, existential statements, and complex relationships.

Example:

- **Propositional logic:** "If it rains, the ground is wet."
  - Inference rule: If `Rain` is true, then `GroundWet` must also be true.
- **First-Order logic:** "For all x, if x is a student, then x studies."
  - Inference can involve variables (`x`), predicates (`Student(x)`), and quantifiers (`∀x`), which makes it more general and applicable to different scenarios.

8.2 Unification and Lifting

Unification is a critical process in first-order logic inference, allowing us to match terms, variables, or predicates in order to apply inference rules.

Unification:

- Unification is the process of making two logical expressions identical by finding a substitution for variables that makes both expressions the same.
- **Example:** Unifying `Loves(John, x)` and `Loves(x, Mary)` would yield the substitution `x = Mary`.

Lifting:

- Lifting is a technique used to extend propositional inference techniques to first-order logic. It involves **lifting** the inference process from propositions to predicates or terms.
- This allows us to infer information over complex structures, such as a person being a parent of another person, rather than just truth values like in propositional logic.

8.3 Forward Chaining

Forward chaining is a data-driven reasoning technique, starting with known facts and applying inference rules to derive new facts until a goal is reached.

1. **Process of Forward Chaining:**
    - Begin with a set of known facts.
    - Continuously apply inference rules to derive new facts.
    - Continue applying rules until the desired conclusion is reached or no new facts can be generated.
2. **Steps in Forward Chaining:**
    - Start with a set of **known facts**.
    - For each fact, apply the available **rules of inference** (e.g., **Modus Ponens**, **Universal Instantiation**).
    - **Store new facts** and repeat the process until a goal is found.
3. **Example:**
    - Known facts:
        1. `Student(John)`
        2. `∀x (Student(x) → Studies(x))`
    - Inference rule: Apply `∀x (Student(x) → Studies(x))` to derive `Studies(John)`.

Advantages of Forward Chaining:

- Efficient when facts are provided and the goal is clear.
- Well-suited for applications like expert systems, where we start with facts and need to derive conclusions.

Disadvantages:

- May require unnecessary computation if the goal is not reached early.

8.4 Backward Chaining

Backward chaining is a goal-driven inference method, where reasoning starts from a goal and works backwards to see if it can be satisfied by known facts.

1. **Process of Backward Chaining:**
   - Start with a goal (what we want to prove or find).
   - Work backwards, finding **subgoals** that must be true in order to reach the goal.
   - Check if the subgoals can be derived from known facts or if more inferences are required.
2. **Steps in Backward Chaining:**
   - Start with a **goal** (e.g., `Studies(John)`).
   - Check if the goal can be directly derived from known facts or by applying inference rules.
   - If not, break the goal into subgoals and recursively attempt to prove them.
3. **Example:**
   - Goal: `Studies(John)`
   - Known fact: `∀x (Student(x) → Studies(x))`
   - Subgoal: `Student(John)`
   - If `Student(John)` is true, then we can conclude `Studies(John)`.

Advantages of Backward Chaining:

- Efficient when a specific goal is given, avoiding unnecessary exploration of facts.
- Used in systems like theorem proving or problem-solving where the goal is predefined.

Disadvantages:

- Can be computationally expensive if the goal has many subgoals or if the search space is large.

8.5 Resolution

Resolution is a complete inference rule used in first-order logic to derive new clauses by finding a contradiction between two clauses.

1. **Process of Resolution:**
   - **Convert the knowledge base** into **conjunctive normal form (CNF)**, where every statement is a conjunction of disjunctions (AND of ORs).
   - **Identify complementary literals** (e.g., `P` and `¬P`).
   - **Resolve** the complementary literals by combining the remaining parts of the two clauses, generating new clauses.
   - **Repeat the resolution** process until either the goal is derived or a contradiction is found.
2. **Example:**
   - Clause 1: `Student(x) ∨ Teacher(x)`
   - Clause 2: `¬Student(x) ∨ Professor(x)`
   - Resolving on `Student(x)`, we get: `Teacher(x) ∨ Professor(x)`
3. **Use of Resolution in Automated Theorem Proving:**
   - Resolution is used in automated theorem provers to prove theorems by repeatedly resolving clauses until a contradiction (or the theorem) is found.

Advantages of Resolution:

- **Sound** and **complete** for first-order logic.
- Can be automated and applied to large knowledge bases.

Disadvantages:

- Can be computationally expensive in practice due to the large number of possible clause combinations.

Exercises

1. **Unification Practice:**
   Given the terms `Loves(John, x)` and `Loves(x, Mary)`, find the unification and the corresponding substitution.
2. **Forward Chaining Example:**
   Given the facts:
   1. `Human(John)`

   2. `∀x (Human(x) → Mortal(x))`
      Apply forward chaining to conclude whether `John` is mortal.
3. **Backward Chaining Example:**
   Using backward chaining, prove the goal `Likes(John, Mary)` given the following rules:
   1. `∀x ∀y (Loves(x, y) → Likes(x, y))`
   2. `Loves(John, Mary)`
4. **Resolution Exercise:**
   Given the clauses:
   1. `¬Likes(John, Mary) ∨ Happy(John)`
   2. `Likes(John, Mary)`
      Apply resolution to derive new information or a contradiction.
5. **Compare Forward and Backward Chaining:**
   Given the same set of facts and a goal, which method would you choose (forward or backward chaining) and why? Provide an example to justify your decision.