AI ASSIGNMENT 8

-Y Sudha Sree (UIN: 662518247)

1. First-Order Logic (FOL) overcomes the shortcomings of Propositional Logic by being more expressive and flexible. While propositional logic deals with simple true/false values for entire statements, it struggles to express relationships between objects, quantify over them, or represent general facts compactly. Below is a breakdown of how First-Order Logic addresses these limitations:

* Expressing Relationships: Propositional logic only deals with atomic propositions like P, Q, or R. It cannot express relationships between objects (e.g., "Sam likes baseball"). Example: In propositional logic, you need separate propositions like L(Sam, Baseball) or L(Ana, Tennis), which become unmanageable for many objects. FOL introduces predicates to express relationships between objects. Example: Likes(Sam, Baseball) means "Sam likes baseball". This allows you to describe richer facts and relationships between objects in a structured manner.
* Handling Quantification (Universal and Existential Quantifiers): Propositional logic cannot express general statements or quantified facts, like "All humans are mortal" or "There exists someone who likes baseball”. To represent these in propositional logic, you'd need to write every instance explicitly (e.g., "Sam is mortal", "Paul is mortal"), which quickly becomes impractical for large sets. FOL allows the use of quantifiers for this purpose:
  + Universal quantifier (∀): "For all". Example: ∀x (Human(x) → Mortal(x)) means "All humans are mortal".
  + Existential quantifier (∃): "There exists". Example: ∃x Likes(x, Baseball) means "There is someone who likes baseball".
* Working with Variables and Functions: Propositional logic cannot handle variables or functions. Each statement must be represented explicitly, which leads to redundancy. Example: If you want to say, "If Ana and Paul both like tennis, they are friends", you'd need separate propositions for every combination. FOL uses variables and functions to represent more complex concepts. Example: Friends(x, y) ← Likes(x, Tennis) ∧ Likes(y, Tennis). Variables x and y make it possible to represent general patterns that apply to multiple individuals, avoiding redundancy.
* Inference with Structured Knowledge: Propositional logic's inference is limited to checking truth values of atomic statements. It cannot infer new information based on structure or relationships between objects. In FOL, you can perform inference using predicates andquantified statements. For example, given:
  + ∀x (Human(x) → Mortal(x)): All humans are mortal.
  + Human(Sam): Sam is a human.

You can infer that Mortal(Sam): Sam is mortal.

* Reusability and Compact Representation: In propositional logic, statements are isolated, making it hard to reuse parts of knowledge efficiently. If new information is added (e.g., a new person), you need to rewrite many propositions. FOL allows for reusable predicates and quantified statements, making knowledge representation more compact and scalable. Example: Instead of listing mortality for every person, the single statement ∀x (Human(x) → Mortal(x)) covers everyone.

First-Order Logic (FOL) enhances expressiveness by allowing relationships, quantification, variables, and structured knowledge. This makes it far more powerful than propositional logic, which is limited to handling simple, isolated statements. As a result, FOL is widely used in domains like artificial intelligence, mathematics, and knowledge representation.

1. You can express universal quantification using existential quantification and negation by applying logical equivalences:

∀x P(x) ≡ ¬∃x ¬P(x)

* ∀x P(x): "For all x, P(x) is true".
* ¬∃x ¬P(x): "It is not true that there exists an x such that P(x) is false".
* If every instance of P(x) is true (universal quantification), then no instance exists where P(x) is false (negated existential quantification).
* Conversely, if there is no instance where P(x) is false, it implies that P(x) holds for all instances.
* Example:
  + Say you want to express: ∀x (Human(x) → Mortal(x))
  + Using existential quantification and negation: ¬∃x ¬(Human(x) → Mortal(x))
  + This expresses the same meaning: "It is not the case that there exists a human who is not mortal".
* This equivalence is a key technique in logic, as it shows that universal quantifiers can be rephrased using existential quantifiers, which is useful in proof strategies and algorithms (e.g., in automated theorem proving).

1. Translating each sentence into First-Order Logic (FOL):
2. Some students took CS411 in Spring2020:
   1. Predicates:
      1. Student(x): x is a student.
      2. Took(x, CS411, Spring2020): x took CS411 in Spring 2020.
   2. Translation: ∃x (Student(x) ∧ Took(x, CS411, Spring2020)). Meaning: There exists a student x such that x took CS411 in Spring 2020.
3. Some students wear a hoodie with UIC logo on it:
   1. Predicates:
      1. Student(x): x is a student.
      2. Wears(x, HoodieWithUICLogo): x wears a hoodie with the UIC logo.
   2. Translation: ∃x (Student(x) ∧ Wears(x, HoodieWithUICLogo)). Meaning: There exists a student x such that x took CS411 in Spring 2020.
4. Something that glitters is not always gold, whereas gold always glitters:
   1. Predicates:
      1. Glitters(x): x glitters.
      2. Gold(x): x is gold.
   2. Translation: ∃x (Glitters(x) ∧ ¬Gold(x)) ∧ ∀y (Gold(y) → Glitters(y)). Meaning: There exists something x that glitters and is not gold, and for all y, if y is gold, then y glitters.
5. No one can win with everyone all the time:
   1. Predicates:
      1. Person(x): x is a person.
      2. WinsWith(x, y): x wins with y (a person).
      3. Time(t): t is a time instance.
   2. Translation: ∀x (Person(x) → ¬∀y ∀t WinsWith(x,y,t)). Meaning: For all persons x, there does not exist a case where x wins with every person y at all times t.
6. All CS courses are difficult, except two:
   1. Predicates:
      1. CS(x): x is a CS course.
      2. Difficult(x): x is difficult.
      3. Exception(x): x is one of the two exceptions.
   2. Translation: ∀x (CS(x) → (Difficult(x) ∨ Exception(x))) ∧ ∃x1 ∃x2 ​(CS(x1​) ∧ CS(x2​) ∧ Exception(x1​) ∧ Exception(x2​) ∧ x1 ≠ x2​). Meaning: For all x, if x is a CS course, then x is either difficult or one of the exceptions, and there exist exactly two exceptions.