L07 - First Order Logic

First Order Logic

Outline

- First-Order Logic (FOL)
  - Syntax
  - Semantics
  - Representation

Objects, Relations and Functions

- First Order Logic resembles natural language in dealing with objects and relations between objects
- Objects: people, houses, colors etc.
  - E.g. John, Red, Ball
- Relations:Verbs or Phrases that relate objects to each other
  - Some relations are **unary** or **properties**: they state some fact about a single object: Round(ball).
  - **n-ary relations** state facts about two or more objects: Married(John, Mary), LargerThan(3,2).

First Order Logic Overview

First Order Logic is designed to resemble natural language when dealing with:

- **Objects** (things in the world)
- **Relations** (connections between objects)

Objects

Objects represent concrete or abstract entities in the world:

- **Examples**: people, houses, colors, etc.
- **Specific instances**: John, Red, Ball
- These are the "nouns" of our logical world

Key Insight

This system allows us to formally represent natural language statements in a logical framework. For instance:

- "The ball is round" → `Round(ball)`
- "John is married to Mary" → `Married(John, Mary)`
- "3 is larger than 2" → `LargerThan(3, 2)`

This forms the foundation for automated reasoning, knowledge representation, and artificial intelligence systems that need to understand and manipulate real-world information.

Relations

Relations are verbs or phrases that connect objects to each other. They come in two types:

1. Unary Relations (Properties)

- Apply to a **single object**
- State some fact or characteristic about one object
- **Example**: `Round(ball)` - states that the ball has the property of being round

2. N-ary Relations

- Apply to **two or more objects**
- State facts about multiple objects and their relationships
- **Examples**:
  - `Married(John, Mary)` - a binary relation stating John and Mary are married
  - `LargerThan(3, 2)` - a binary relation comparing two numbers

Key Insight

Ontological Commitment

- What the language assumes about the nature of reality
- FOL vs Propositional logic
  - Propositional logic assumes that there are facts that either hold or do not hold
  - FOL assumes there are objects in the world, and there are relations among them that do or do not hold
- Temporal logic
  - Facts hold at particular times
- Higher-order logic
  - Relations and functions are also objects
  - More expressive than FOL

What is Ontological Commitment?

Ontological commitment refers to what assumptions a logical language makes about what exists in reality and how reality is structured.

Comparison of Different Logic Systems:

Propositional Logic

- **Assumption**: Reality consists of facts that are either true or false
- **Simple binary view**: Things either hold or don't hold
- **No structure**: Doesn't recognize objects or relationships between them

First Order Logic (FOL)

- **Assumption**: Reality contains:
  - **Objects** that exist in the world
  - **Relations** among these objects that either hold or don't hold
- **More structured**: Recognizes entities and their interconnections
- **Example**: Can represent "John loves Mary" as `Loves(John, Mary)`

Temporal Logic

- **Assumption**: Facts and relationships change over time
- **Time-sensitive**: Facts hold at particular moments
- **Example**: "John was married in 2020" vs "John is married now"

Higher-Order Logic

- **Assumption**: Relations and functions are themselves objects
- **Most expressive**: Can reason about properties of properties
- **Meta-level reasoning**: Can have relations between relations
- **More powerful than FOL**: Can express concepts that FOL cannot

Key Insight

Each logic system makes increasingly complex assumptions about reality:

- **Propositional**: Just facts
- **FOL**: Facts about objects and their relationships
- **Temporal**: Facts that vary over time
- **Higher-order**: Facts about facts (meta-reasoning)

The choice of logical framework determines what kinds of knowledge you can represent and reason about in your system.

Epistemological Commitments

- Possible states of knowledge a logic allows wrt each fact

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Models of FOL

- Domain of a model- set of objects it contains
- Objects are also called domain elements
- Relations are sets of tuples of objects that are related
- Functions should be total functions

Domain of a Model

- **Definition**: The set of all objects that exist in the model
- **What it contains**: All the entities/things that the model recognizes
- **Alternative terminology**: Objects in the domain are also called **domain elements**

Objects/Domain Elements

- These are the individual entities that populate the model's universe
- They correspond to the "things" that exist in the world being modeled
- Examples might include: specific people, numbers, physical objects, etc.

Relations

- **Structure**: Relations are represented as **sets of tuples** of objects
- **What this means**: A relation connects objects together in specific patterns
- **Example**: If we have a "Married" relation, it would be a set of pairs like {(John, Mary), (Bob, Alice)} indicating which people are married to each other

Functions

- **Requirement**: Functions must be **total functions**
- **What "total" means**: The function must be defined for every possible input from the domain
- **No gaps**: There cannot be any domain elements for which the function is undefined

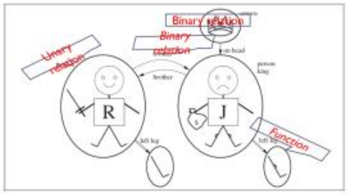
Key Insight

This describes the mathematical foundation of how FOL models represent real-world scenarios. A model provides:

1. A universe of objects (domain)
2. Specific relationships between those objects (relations as tuple sets)
3. Operations that map objects to other objects (total functions)

This formal structure allows FOL to precisely capture and reason about complex real-world situations in a mathematically rigorous way.

Models for FOL



Symbols

- Constant symbols
  - For objects e.g. Richard, John
- Predicate symbols
  - For relations e.g. Brother, OnHead, Person
- Function symbols
  - For functions e.g. LeftLeg

Specify arity

1. Constant Symbols

- **Purpose:** Represent specific objects in the domain
- **Examples:** Richard, John
- **What they do:** These are names for particular, individual entities
- **Think of them as:** Proper nouns that refer to specific things

2. Predicate Symbols

- **Purpose:** Represent relations between objects
- **Examples:** Brother, OnHead, Person
- **What they do:** Express relationships or properties that can be true or false
- **Usage examples:**
  - `Brother(John, Richard)` - "John is a brother of Richard"
  - `Person(John)` - "John is a person"
  - `OnHead(Hat, John)` - "Hat is on John's head"

3. Function Symbols

- **Purpose:** Represent functions that map objects to other objects
- **Examples:** LeftLeg
- **What they do:** Take one or more objects as input and return another object
- **Usage example:** `LeftLeg(John)` - returns the object that is John's left leg

Interpretations

- An intended interpretation specifies
  - Which object is referred by a constant symbol
    - *Richard* refers to Richard the Lionheart
  - Which relation is referred by a predicate symbol
    - *Brother* refers to brotherhood relation
  - Which function is referred by a function symbol
    - *leftLeg* refers to the left leg function
- Truth should be defined in terms of all possible models and all possible interpretations

What is an Intended Interpretation?

An intended interpretation specifies the **concrete meaning** that we assign to abstract logical symbols. It creates the bridge between formal logic and real-world entities.

Components of an Interpretation:

1. Constant Symbol Mapping

- **What it specifies:** Which real-world object each constant symbol refers to
- **Example:** The symbol `Richard` refers to "Richard the Lionheart"
- **Purpose:** Connects abstract names to specific individuals

2. Predicate Symbol Mapping

- **What it specifies:** Which real-world relation each predicate symbol represents
- **Example:** The symbol `Brother` refers to the "brotherhood relation"
- **Purpose:** Defines what relationships the predicate actually means

3. Function Symbol Mapping

- **What it specifies:** Which real-world function each function symbol represents
- **Example:** The symbol `leftLeg` refers to "the left leg function"
- **Purpose:** Defines what operation or mapping the function performs

Truth and Multiple Interpretations

The bottom point highlights a crucial concept:

- **Truth is relative:** A logical statement's truth depends on the interpretation
- **Multiple possibilities:** The same logical formula can have different truth values under different interpretations
- **Model theory:** Truth must be evaluated across all possible models and interpretations

Key Insight

Interpretations allow the same logical framework to represent different scenarios. For example:

- `Brother(Richard, John)` could mean different things depending on whether we're talking about historical figures, fictional characters, or mathematical objects
- The logical structure remains the same, but the meaning changes based on the interpretation

This flexibility makes FOL a powerful tool for representing diverse domains of knowledge.

Syntax of FOL: basic elements

- Constants Dan, UoM, 5, ...
- Predicates Brother, >, Before...
- Functions Sqrt, Length, ...
- Variables x, y, a, b,...
- Connectives ⇒, ∧, ∨, ⇔
- Equality =
- Quantifiers ∀, ∃

Terms

- Logical expressions that refer to objects
  - *John*
  - *LeftLeg(John)*
  - *LeftLeg(x)*

Syntax of FOL: Sentence

- Sentence → AtomicSentence
  - | Sentence *Connective* Sentence
  - | Quantifier Variable, … Sentence
  - | ¬Sentence
- AtomicSentence →
  - Predicate(Term, …) | Term = Term
- Term →
  - Function(Term, …) | Constant | Variable
- Connective → ⇒ | ∧ | ∨ | ⇔
- Quantifier → ∀ | ∃

Sentence Construction

```
Sentence → AtomicSentence
         | Sentence Connective Sentence
         | Quantifier Variable, ... Sentence
         | ¬Sentence
```

A **Sentence** can be formed in four ways:

1. **AtomicSentence:** The simplest form
2. **Compound Sentence:** Two sentences connected by a logical connective
3. **Quantified Sentence:** A quantifier applied to variables and a sentence
4. **Negated Sentence:** A sentence with negation (¬)

AtomicSentence Structure

```
AtomicSentence → Predicate(Term, ...) | Term = Term
```

AtomicSentences are the building blocks and can be:

- **Predicate applications:** Like `Brother(John, Richard)`
- **Equality statements:** Like `x = y`

Term Construction

```
Term → Function(Term, ...) | Constant | Variable
```

Terms represent objects and can be:

- **Function applications:** Like `LeftLeg(John)`
- **Constants:** Like `Richard`, `John`
- **Variables:** Like `x`, `y`

Logical Connectives

```
Connective → ⇒ | ∧ | ∨ | ⇔
```

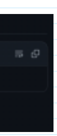
The logical operators are:

- **⇒:** Implication (if-then)
- **∧:** Conjunction (and)
- **∨:** Disjunction (or)
- **⇔:** Biconditional (if and only if)

Quantifiers

```
Quantifier → ∀ | ∃
```

- **∀:** Universal quantifier ("for all")
- **∃:** Existential quantifier ("there exists")



Sentences

- Atomic sentences
  - Formed by a predicate symbol followed by a parenthesized list of terms
    - *Brother(Richard, John)*
  - May have complex terms as arguments
    - *Married(Father(Richard), Mother(John))*
- Complex sentences
  - Constructed by using logical connectives
  - Similar to sentences in propositional logic
    - ¬Brother(LeftLeg(Richard), John)
    - Brother(Richard, John) ∧ Brother(John, Richard)
    - King(Richard) ∨ King(John)
    - ¬King(Richard) ⇒ King(John)

Atomic Sentences

These are the basic building blocks of FOL:

Basic Structure

- **Format**: Predicate symbol followed by a parenthesized list of terms
- **Example**: `Brother(Richard, John)`
- **Meaning**: "Richard is a brother of John"

Complex Terms as Arguments

- Atomic sentences can have complex terms (like function applications) as arguments
- **Example**: `Married(Father(Richard), Mother(John))`
- **Meaning**: "Richard's father is married to John's mother"
- This shows how functions can be nested within predicates

Complex Sentences

These are built from atomic sentences using logical operators:

Construction Method

- **Built using**: Logical connectives (similar to propositional logic)
- **Components**: Combine atomic sentences with logical operators

Examples of Complex Sentences

1. **Negation**: `¬Brother(LeftLeg(Richard), John)`
   - "It is not the case that Richard's left leg is a brother of John"
2. **Conjunction**: `Brother(Richard, John) ∧ Brother(John, Richard)`
   - "Richard is a brother of John AND John is a brother of Richard"
3. **Disjunction**: `King(Richard) ∨ King(John)`
   - "Richard is a king OR John is a king"
4. **Biconditional**: `¬King(Richard) ⇔ King(John)`
   - "Richard is not a king IF AND ONLY IF John is a king"

Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: "for all" ∀
- Existential: "there exists" ∃

Universal quantification

∀ <*variables*> <*sentence*>

Everyone at UoM is smart:

∀x At(x, UoM) ⇒ Smart(x)

∀x *P* is true in a model *m* iff *P* is true with *x* being ***each*** possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

At(Dan, UoM) ⇒ Smart(Dan)
∧ At(Richard, UoM) ⇒ Smart(Richard)
∧ At(Ben, UoM) ⇒ Smart(Ben)
∧ ...

A common mistake to avoid

- Typically, ⇒ is the main connective with ∀
  - A universal quantifier is also equivalent to a set of implications over all objects
- Common mistake: using ∧ as the main connective with ∀:

  ∀x At(x, UoM) ∧ Smart(x)

  means "Everyone is at UoM and everyone is smart"

  Leads to overly strong statements

Existential quantification

∃ <*variables*> <*sentence*>

Someone at UoM is smart:

∃*x* At(x, UoM) ∧ Smart(x)

∃*x* *P* is true in a model *m* iff *P* is true with *x* being ***some*** possible object in the model

- Roughly speaking, equivalent to the disjunction of instantiations of *P*

At(Dan, UoM) ∧ Smart(Dan)
∨ At(Richard, UoM) ∧ Smart(Richard)
∨ At(Ben, UoM) ∧ Smart(Ben)
∨ ...

Another common mistake to avoid

- Typically, ∧ is the main connective with ∃
- Common mistake: using ⇒ as the main connective with ∃:

  ∃*x* At(x, UoM) ⇒ Smart(x)

  is true even for someone who is not at UoM!

Properties of quantifiers

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$
$\exists x\ \exists y$ is the same as $\exists y\ \exists x$

$\exists x\ \forall y$ is *not* the same as $\forall y\ \exists x$
$\exists x\ \forall y$ Loves(x,y)
- "There is a person who loves everyone in the world"

$\forall y\ \exists x$ Loves(x,y)
- "Everyone in the world is loved by at least one person"

- Quantifier duality: each can be expressed using the other
  $\forall x$ Likes(x,IceCream)
  $\neg \exists x\ \neg$Likes(x,IceCream)
  $\exists x$ Likes(x,Broccoli)
  $\neg \forall x\ \neg$Likes(x,Broccoli)

De Morgan Rules

- Universal quantifier is a conjunction
- Existential quantifier is a disjunction

$$
\begin{array}{llll}
\forall x\ \neg P \equiv \neg\exists x\ P & & \neg(P \vee Q) & \equiv \neg P \wedge \neg Q \\
\neg\forall x\ P \equiv \exists x\ \neg P & & \neg(P \wedge Q) & \equiv \neg P \vee \neg Q \\
\forall x\ P \equiv \neg\exists x\ \neg P & & P \wedge Q & \equiv \neg(\neg P \vee \neg Q) \\
\exists x\ P \equiv \neg\forall x\ \neg P & & P \vee Q & \equiv \neg(\neg P \wedge \neg Q)
\end{array}
$$

Equality

- *term*$_1$ = *term*$_2$ is true under a given interpretation if and only if *term*$_1$ and *term*$_2$ refer to the same object

- E.g., definition of *Sibling* in terms of *Parent*:
  $\exists$ x,y Brother(x, Richard) $\wedge$ Brother(y, Richard) $\wedge$ $\neg$(x=y)

Using First Order Logic

- Assertions
  - Sentences that are added to the KB using TELL
    - *TELL(KB, King(John))*
    - *TELL(KB, $\forall$x King(x) $\Rightarrow$ Person(x))*
- Queries (goals)
  - Ask questions of the KB using ASK
    - *ASK(KB, King(John))*
  - Quantified queries (Answer is a substitution or a binding list)
    - *ASK(KB, $\exists$x Person(x))*

Example: The Royal Kinship Domain

- Includes facts s.a.
  - Elizabeth is the mother of Charles
  - Charles is the father of William
  - William is the husband of Kate
  - One's grandmother is the mother of one's parent
- Unary predicates
  - Male, Female
- Binary predicates
  - Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, GrandChild, Cousin, Aunt, Uncle

Axioms

- Provide the basic factual information from which useful conclusions can be derived
- One's mother is one's female parent
  - *$\forall$m,c Mother(c) = m $\Leftrightarrow$ (Female(m) $\wedge$ Parent(m,c))*
- One's husband is one's male spouse
  - *$\forall$w, h Husband(h, w) $\Leftrightarrow$ Male(h) $\wedge$ Spouse(h, w)*
- Male and female are disjoint categories
  - *$\forall$x Male(x) $\Leftrightarrow$ $\neg$Female(x)*
- Parent and child are inverse relations
  - *$\forall$p, c Parent(p, c) $\Leftrightarrow$ Child(c, p)*

Axioms

- The above axioms are also definitions
- Some axioms are just plain facts
  - Male(Harry)
- Some sentences are theorems, which are entailed by axioms
  - $\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$
    - *Logically follows from the axiom 'a sibling is another child of one's parent'*
    - $\forall x, y\ Sibling(x,y) \Leftrightarrow \neg(x=y) \land \exists p\ Parent(p,x) \land Parent(p,y)$

25

Summary

- First-order logic:
  - Much more expressive than propositional logic
  - Allows objects and relations as semantic primitives
  - Universal and existential quantifiers
  - Syntax

26