The most widely used formal logic method is **FIRST-ORDER PREDICATE LOGIC**

Reference: Chapter Two The predicate Calculus
Luger’s Book
Examples included from Norvig and Russel.
First-order logic

- Whereas propositional logic assumes the world contains **facts**, 
- first-order logic (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, colors, baseball games, wars, …
  - **Relations**: red, round, prime, brother of, bigger than, part of, comes between, …
  - **Functions**: father of, best friend, one more than, plus, …
Syntax of FOL: Basic elements

- Constants: john, 2, lums,...
- Predicates: brother, >,...
- Functions: sqrt, leftsideOf,...
- Variables: X, Y, A, B,...
- Connectives: ¬, ⇒, ∧, ∨, ⇔
- Equality: =
- Quantifiers: ∀, ∃
Truth in first-order logic

- Sentences are true with respect to a model and an interpretation.

- Model contains objects (domain elements) and relations among them.

- Interpretation specifies referents for:
  - Constant symbols → objects
  - Predicate symbols → relations
  - Function symbols → functional relations

- An atomic sentence $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is true iff the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by $\text{predicate}$. 
Alphabets-I

Predicates, variables, functions, constants, connectives, quantifiers, and delimiters

**Constants:** (first letter small)
- bLUE: a color
- sanTRO: a car
- crow: a bird

**Variables:** (first letter capital)
- Dog: an element that is a dog, but unspecified
- Color: an unspecified color
Alphabets-II

Function:
Maps Sentences to Objects

Ali is father of Babar

father(babar) = ali
father_of(baber) = ali

• Interpretation has to be very clear.
• If you write father(baber), the answer should be ali
• For the above functions the *arity is 1*

(number of arguments to the function)
Alphabets-II

Functions:
1) shahid likes zahid   \( \text{likes}(\text{shahid}) = \text{zahid} \)
2) atif likes abid     \( \text{likes}(\text{atif}) = \text{abid} \)
3) Constants to Variables \( \text{likes}(X) = Y \)

\( \{X, Y\} \) have two possible BINDINGS

\( \{X, Y\} \) could be \{shahid, zahid\}
Or
\( \{X, Y\} \) could be \{atif, abid\}

Substitutions:
For 1 to be true:
\( \{\text{shahid}/X, \text{zahid}/Y\} \)

For 2 to be true:
\( \{\text{atif}/X, \text{abid}/Y\} \)
Alphabets-II

Predicate

Maps Sentences to Truth Values (True/False)

1) Shahid is student  \text{student}(shahid)
2) Sana is a girl  \text{girl}(sana)
3) Father of baber is elder than Hamza  
   \text{elder}([\text{father}(\text{babar}), \text{hamza}])

\textbf{For 1 and 2 arity is 1 and for 3 the arity is 2}
Alphabets-II

**Predicate**

1) Shahid is a good student
   student(shahid,good) or good_student(shahid)

2) Sana is a friend of Saima, Sana and Saima both are girls
   friend_of(sana,saima) ^ girl(sana) ^ girl(saima)

3) Bill helps Fred
   helps(bill,fred)
Atomic sentences

Atomic sentence = \textit{predicate} \ (\textit{term}_1,\ldots,\textit{term}_n) \\
or \textit{term}_1 = \textit{term}_2

\text{term} = \textit{function} \ (\textit{term}_1,\ldots,\textit{term}_n) \\
or \textit{constant} \\
or \textit{variable}

- \textit{brother}(john,richard)
- \textit{greater}(length(leftsideOf(squareA)), length(leftsideOf(squareB)))
- \textit{>(length(leftsideOf(squareA)), length(leftsideOf(squareB)))}

\textit{Functions cannot be atomic sentences}
Alphabets-III

Connectives:
- $\wedge$ and
- $\lor$ or
- $\neg$ not
- $\rightarrow$ Implication

Quantification
- All persons can see
- There is a person who cannot see

Universal quantifiers $\forall$ (ALL)
Existential quantifiers $\exists$ (There exists)
Complex sentences

- Complex sentences are made from atomic sentences using connectives
  \[-S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2,\]

\[
\text{sibling(ali,hamza)} \Rightarrow \text{sibling(hamza,ali)}
\]

\[
> (1,2) \lor \leq (1,2) \quad (1 \text{ is greater than } 2 \text{ or less than equal to } 2)
\]

\[
> (1,2) \land \neg > (1,2) \quad (1 \text{ is greater than } 2 \text{ and is not greater than equal to } 2)
\]
Examples

My house is a blue, two-story, with red shutters, and is a corner house

\text{blue(my-house)} \land \text{two-story(my-house)} \land \text{red-shutters(my-house)} \land \text{corner(my-house)}

Ali bought a scooter or a car

\text{bought(ali, car)} \lor \text{bought(ali, scooter)}

IF fuel, air and spark are present the fuel will combust

\text{present(spark)} \land \text{present(fuel)} \land \text{present(air)} \rightarrow \text{combustion(fuel)}
Universal quantification

- $\forall \langle \text{variables} \rangle \langle \text{sentence} \rangle$
  Everyone at LUMS is smart:
  $\forall X \ \text{at}(X, \text{lums}) \Rightarrow \text{smart}(X)$

- $\forall X \ P$ 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 $P$

  $\text{at}(\text{rabia}, \text{lums}) \Rightarrow \text{smart(rabia)}$
  $\land \text{at}(\text{shahid}, \text{lums}) \Rightarrow \text{smart(shahid)}$
  $\land \text{at}(\text{lums}, \text{lums}) \Rightarrow \text{smart(lums)}$
  $\land \ldots$
Examples

All people need air

\[ \forall X[\text{person}(X) \implies \text{need\_AIR}(X)] \]

The owner of the car also owns the boat

[\text{owner}(X, \text{car}) \land \text{car}(X, \text{boat})]

Formulate the following expression in the PC:

“Ali is a computer science student but not a pilot or a football player”

\[ \text{cs\_STUDENT}(\text{ali}) \land (\neg \text{pilot}(\text{ali}) \lor \neg \text{ft\_PLAYER}(\text{ali})) \]
Examples

Restate the sentence in the following way:
1. Ali is a computer science (CS) student
2. Ali is not a pilot
3. Ali is not a football player

\[
\text{cs\_student(ali)} \land \neg \text{pilot(ali)} \land \neg \text{football\_player(ali)}
\]
Examples

Studying fuzzy systems is exciting and applying logic is great fun if you are not going to spend all of your time slaving over the terminal

\( \forall X(\neg \text{slave\_terminal}(X) \rightarrow [fs\_eciting(X)^\land \text{logic\_fun}(X)]) \)

Every voter either favors the amendment or despises it

\( \forall X[\text{voter}(X) \rightarrow [\text{favor}(X, \text{amendment}) \lor \neg \text{favor}(X, \text{amendment}) \land \neg \text{despise}(X, \text{amendment})] \)

(this part simply endorses the statement, may not be required)
Undecidable Predicate

• For which exhaustive testing is required
• Example:
  $\forall X \text{ likes}(zahra, X)$
• This sentence is computationally impossible to calculate
• Scope of problem domain is to be limited to remove this problem,
  – i.e., $X$ is a variable representing final year female student in the AI class, compared to an $X$ representing all the people in the city of Lahore
Robotic Arm Example

• Represent the initial details of the system
• Generate sentences of descriptive and or implicative nature
• Modify the facts using new sentences
Example: Robotic Arm

- Represent the initial details of the systems

```
on(a,b)
on(c,d)
ontable(b)
ontable(d)
clear(a)
clear(c)
hand_empty
```
FOL

**Goal:**
To pick a block and place it over another block

**Define predicate:** \( \text{stack}_\text{on}(X,Y) \)

**General sentence:**
hand_empty \( ^ \) clear (X) \( ^ \) clear (Y) \( ^ \) pick (X) \( ^ \) put_on (X,Y) \( \Rightarrow \) stack_on (X,Y)

---

**Conditions** --- **Conclusions**

What could the conditions be?

- hand_empty
- clear (X)
- clear (Y)
- pick (X)
- put_on (X,Y)
Goal:
To pick a block and place it over another block

\[
\text{hand_empty} \land \text{clear}(X) \land \text{clear}(Y) \land \text{pick}(X) \land \text{put_on}(X,Y) \rightarrow \text{stack_on}(X,Y)
\]

Semantically more correct:

\[
\text{hand_empty} \land \text{clear}(X) \rightarrow \text{pick}(X)
\]
\[
\text{clear}(Y) \land \text{pick}(X) \rightarrow \text{put_on}(X,Y)
\]
\[
\text{put_on}(X,Y) \rightarrow \text{stack_on}(X,Y)
\]

hand_empty could be written as empty(hand), if hand_empty is in the knowledge base, then hand is empty otherwise false.

put_on(X,Y) → stack_on(X,Y) is in fact equivalence
Example: Robotic Arm

- Modify details of the systems

```
on(b,a)
on(c,d)
tonable(b)
tonable(d)
clear(a)
clear(c)
hand_empty
```

```
on(b,a)
on(c,d)
on(e,a)
tonable(b)
tonable(d)
clear(c)
clear(e)
hand_empty
```
Models for FOL: Example
A common mistake to avoid

- **Represent:** Everyone at LUMS is smart

\[ \forall X \ (\text{at}(X, \text{lums}) \land \text{smart}(X)) \]

\[ \forall X \ (\text{at}(X, \text{lums}) \Rightarrow \text{smart}(X)) \]

- **Common mistake:**

  using \( \land \) as the main connective with \( \forall \): means

  “Everyone is at LUMS and everyone is smart”
  “Everyone at LUMS is smart”

- **Typically,**

  \( \Rightarrow \) is the main connective with \( \forall \)
Existential quantification

• $\exists<\text{variables}> <\text{sentence}>$

• Someone at LUMS is smart:
  • $\exists X \text{ at}(X,\text{lums}) \land \text{smart}(X)$

• $\exists X \ P$ 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 $P$

  $\text{at}(\text{sana},\text{lums}) \land \text{smart}(\text{sana})$
  $\lor \text{at}(\text{bashir},\text{lums}) \land \text{smart}(\text{bashir})$
  $\lor \text{at}(\text{lums},\text{lums}) \land \text{smart}(\text{lums})$
  $\lor \ldots$
Another common mistake to avoid

- Typically, \( \land \) is the main connective with \( \exists \)

- **Common mistake**: using \( \Rightarrow \) as the main connective with \( \exists \):

  \[ \exists X \; \text{at}(X, \text{lums}) \Rightarrow \text{smart}(X) \]

  is true if there is anyone who is not at LUMS!

- **Even if the antecedent is false the sentence can still be true** (see the truth table of implication).
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($X$,car) $\rightarrow \neg \exists X \neg$ likes($X$,car)
  - $\exists X$ likes($X$,bread) $\rightarrow \neg \forall X \neg$ likes($X$,bread)
Equality

- \( \text{term}_1 = \text{term}_2 \) is true under a given interpretation if and only if \( \text{term}_1 \) and \( \text{term}_2 \) refer to the same object

- **Sibling** in terms of **Parent**:
  \[
  \forall X,Y \ \text{sibling}(X,Y) \leftrightarrow \\
  [\neg(X = Y) \land \exists M,F \neg(M = F) \land \\
  \text{parent}(M,X) \land \text{parent}(F,X) \land \\
  \text{parent}(M,Y) \land \text{parent}(F,Y)]
  \]
Using FOL

The kinship domain:

- Brothers are siblings
  \[ \forall X, Y \; \text{brother}(X, Y) \iff \text{sibling}(X, Y) \]
- One's mother is one's female parent
  \[ \forall M, C \; \text{mother}(C) = M \iff (\text{female}(M) \land \text{parent}(M, C)) \]
- “Sibling” is symmetric
  \[ \forall X, Y \; \text{sibling}(X, Y) \iff \text{sibling}(Y, X) \]
Rules: Wumpus world

- **Perception**
  - $\forall T, S, B \text{ percept}([S, B, \text{glitter}], T) \Rightarrow \text{glitter}(T)$

- **Reflex**
  - $\forall T \text{ glitter}(T) \Rightarrow \text{bestAction}($grab$, T)$
Deducing Squares/Properties

What are Adjacent Squares

\[ \forall X,Y,A,B \text{ adjacent}([X,Y],[A,B]) \iff [A,B] \in \{[X+1,Y], [X-1,Y],[X,Y+1],[X,Y-1]\} \]

Properties of squares:

\[ \forall S,T \text{ at}(\text{agent},S,T) \land \text{breeze}(T) \Rightarrow \text{breezy}(S) \]

Squares are breezy near a pit:

Diagnostic rule---infer cause from effect

\[ \forall S \text{ breezy}(S) \Rightarrow \text{adjacent}(R,S) \land \text{pit}(R) \]

Causal rule---infer effect from cause

\[ \forall R \text{ pit}(R) \Rightarrow [\forall S \text{ adjacent}(R,S) \Rightarrow \text{breezy}(S)] \]
Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base
The electronic circuits domain

One-bit full adder

![One-bit full adder diagram]
The electronic circuits domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)

2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates

3. Decide on a vocabulary
   - Alternatives:
     type(x₁) = xor
     type(x₁, xor)
     xor(x₁)
4. Encode general knowledge of the domain

- \( \forall T_1, T_2 \text{ connected}(T_1, T_2) \Rightarrow \text{signal}(T_1) = \text{signal}(T_2) \)
- \( \forall T \text{ signal}(T) = 1 \lor \text{signal}(T) = 0 \)
- \( 1 \neq 0 \)
- \( \forall T_1, T_2 \text{ connected}(T_1, T_2) \Rightarrow \text{connected}(T_2, T_1) \)
- \( \forall G \text{ type}(G) = \text{OR} \Rightarrow \text{signal}(\text{out}(1,G)) = 1 \iff \exists N \text{ signal}(\text{in}(N,G)) = 1 \)
- \( \forall G \text{ type}(G) = \text{AND} \Rightarrow \text{signal}(\text{out}(1,G)) = 0 \iff \exists N \text{ signal}(\text{in}(N,G)) = 0 \)
- \( \forall G \text{ type}(G) = \text{XOR} \Rightarrow \text{signal}(\text{out}(1,G)) = 1 \iff \text{signal}(\text{in}(1,G)) \neq \text{signal}(\text{in}(2,G)) \)
- \( \forall G \text{ type}(G) = \text{NOT} \Rightarrow \text{signal}(\text{out}(1,G)) \neq \text{signal}(\text{in}(1,G)) \)
The electronic circuits domain

5. Encode the specific problem instance

\[
\begin{align*}
\text{type}(x_1) &= \text{xor} & \text{type}(x_2) &= \text{xor} \\
\text{type}(a_1) &= \text{and} & \text{type}(a_2) &= \text{and} \\
\text{type}(o_1) &= \text{or}
\end{align*}
\]

\[
\begin{align*}
\text{connected}(\text{out}(1,x_1),\text{in}(1,x_2)) & & \text{connected}(\text{in}(1,c_1),\text{in}(1,x_1)) \\
\text{connected}(\text{out}(1,x_1),\text{in}(2,a_2)) & & \text{connected}(\text{in}(1,c_1),\text{in}(1,a_1)) \\
\text{connected}(\text{out}(1,o_2),\text{in}(1,o_1)) & & \text{connected}(\text{in}(2,c_1),\text{in}(2,x_1)) \\
\text{connected}(\text{out}(1,a_1),\text{in}(2,o_1)) & & \text{connected}(\text{in}(2,c_1),\text{in}(2,a_1)) \\
\text{connected}(\text{out}(1,x_2),\text{out}(1,c_1)) & & \text{connected}(\text{in}(3,c_1),\text{in}(2,x_2)) \\
\text{connected}(\text{out}(1,o_1),\text{out}(2,c_1)) & & \text{connected}(\text{in}(3,c_1),\text{in}(1,a_2))
\end{align*}
\]
The electronic circuits domain

6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

\[ \exists I_1, I_2, I_3, O_1, O_2 \quad \text{signal}(\text{in}(1,c_1)) = I_1 \land \text{signal}(\text{in}(2,c_1)) = I_2 \land \text{signal}(\text{in}(3,c_1)) = I_3 \land \text{signal}(\text{out}(1,c_1)) = O_1 \land \text{signal}(\text{out}(2,o_1)) = O_2 \]

7. Debug the knowledge base

May have omitted assertions like \( 1 \neq 0 \)
Summary

• First-order logic:
  – objects and relations are semantic primitives
  – syntax: constants, functions, predicates, equality, quantifiers

• Increased expressive power: sufficient to define wumpus world
Operations

- **Unification**: Algorithm for determining the substitutions needed to make two predicate calculus expressions match
- **Skolemization**: A method of removing or replacing existential quantifiers
- **Composition**: If $S$ and $S'$ are two substitutions sets, then the composition of $S$ and $S'$ ($SS'$) is obtained by applying the elements of $S$ to the elements of $S'$ and finally adding the results