First-order logic

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Limitations of propositional logic

World we want to represent and reason about consists of a number of objects with variety of properties and relations among them

Propositional logic:
• Represents statements about the world without reflecting this structure and without modeling these entities explicitly

Consequence:
• some knowledge is hard or impossible to encode in the propositional logic.
• Two cases that are hard to represent:
  – Statements about similar objects, relations
  – Statements referring to groups of objects.
First-order logic (FOL)

- More expressive than **propositional logic**

- **Eliminates deficiencies of PL by:**
  - Representing objects, their properties, relations and statements about them;
  - Introducing variables that refer to an arbitrary objects and can be substituted by a specific object
  - Introducing quantifiers allowing us to make statements over groups objects without the need to represent each of them separately

---

**Logic**

Logic is defined by:

- **A set of sentences**
  - A sentence is constructed from a set of primitives according to syntax rules.

- **A set of interpretations**
  - An interpretation gives a semantic to primitives. It associates primitives with objects, values in the real world.

- **The valuation (meaning) function** $V$
  - Assigns a truth value to a given sentence under some interpretation

$$V : \text{sentence} \times \text{interpretation} \rightarrow \{\text{True, False}\}$$
First-order logic. Syntax.

**Term** - syntactic entity for representing objects

**Terms in FOL:**
- **Constant symbols**: represent specific objects
  - E.g. *John, France, car89*
- **Variables**: represent objects of a certain type (type = domain of discourse)
  - E.g. *x, y, z*
- **Functions** applied to one or more terms
  - E.g. *father-of (John)*
    
    \[
    \text{father-of(father-of(John))}
    \]

---

First order logic. Syntax.

**Sentences in FOL:**
- **Atomic sentences:**
  - A **predicate symbol** applied to 0 or more terms
    
    **Examples**:
    
    \[
    \text{Red(car12),}
    \text{Sister(Amy, Jane);}
    \text{Manager(father-of(John));}
    \]
    
    \[
    \text{t1 = t2 equivalence of terms}
    \]
    
    **Example**:
    
    \[
    \text{John = father-of(Peter)}
    \]
First order logic. Syntax.

Sentences in FOL:
- **Complex sentences:**
  - Assume \( \phi, \psi \) are sentences in FOL. Then:
    - \( (\phi \land \psi) \) (\( \phi \lor \psi \)) (\( \phi \Rightarrow \psi \)) (\( \phi \Leftrightarrow \psi \)) \( \neg \psi \)
    - and
    - \( \forall x \phi \) \( \exists y \phi \)
  - are sentences

Symbols \( \exists, \forall \)
- stand for the **existential** and the **universal** quantifier

Semantics. Interpretation.

An interpretation \( I \) is defined by a **mapping** to the **domain of discourse** \( D \) or **relations on** \( D \)
- **domain of discourse:** a set of objects in the world we represent and refer to;

**An interpretation \( I \) maps:**
- Constant symbols to objects in \( D \)
  \( I(John) = \) 
- Predicate symbols to relations, properties on \( D \)
  \( I(brother) = \{ \langle \overset{\text{John}}{\text{John}} \rangle; \langle \overset{\text{John}}{\text{John}} \rangle; \ldots \} \)
- Function symbols to functional relations on \( D \)
  \( I(father-of) = \{ \langle \overset{\text{John}}{\text{John}} \rangle \rightarrow \overset{\text{Mary}}{\text{Mary}}; \langle \overset{\text{John}}{\text{John}} \rangle \rightarrow \overset{\text{Mary}}{\text{Mary}}; \ldots \} \)
Semantics of sentences.

Meaning (evaluation) function:

\[ V : \text{sentence} \times \text{interpretation} \rightarrow \{\text{True}, \text{False}\} \]

A predicate `predicate(term-1, term-2, term-3, term-n)` is true for the interpretation `I`, iff the objects referred to by `term-1, term-2, term-3, term-n` are in the relation referred to by `predicate`.

\[ I(John) = \begin{cases} \Box_1 \\ \Box_2 \end{cases} \quad I(Paul) = \begin{cases} \Box_1 \\ \Box_2 \end{cases} \]

\[ I(\text{brother}) = \left\{ \langle \begin{cases} \Box_1 \\ \Box_2 \end{cases}, \begin{cases} \Box_1 \\ \Box_2 \end{cases} \rangle; \langle \begin{cases} \Box_1 \\ \Box_2 \end{cases}, \begin{cases} \Box_1 \\ \Box_2 \end{cases} \rangle; \ldots \right\} \]

\[ \text{brother}(John, Paul) = \begin{cases} \Box_1 \\ \Box_2 \end{cases} \quad \text{in } I(\text{brother}) \]

\[ V(\text{brother}(John, Paul), I) = \text{True} \]

Semantics of sentences.

- **Equality**
  \[ V(\text{term-1} = \text{term-2}, I) = \text{True} \]
  \[ \text{If} \quad I(\text{term-1}) = I(\text{term-2}) \]

- **Boolean expressions: standard**
  E.g. \[ V(\text{sentence-1} \lor \text{sentence-2}, I) = \text{True} \]
  \[ \text{If} \quad V(\text{sentence-1}, I) = \text{True} \quad \text{or} \quad V(\text{sentence-2}, I) = \text{True} \]

- **Quantifications**
  \[ V(\forall x \phi, I) = \text{True} \quad \text{substitution of } x \text{ with } d \]
  \[ \text{If} \quad \text{for all } d \in D \quad V(\phi, I[x/d]) = \text{True} \]

  \[ V(\exists x \phi, I) = \text{True} \]
  \[ \text{If} \quad \text{there is a } d \in D \quad \text{s.t.} \quad V(\phi, I[x/d]) = \text{True} \]
Representing knowledge in FOL

Example:

**Kinship domain**

- **Objects:** people
  
  \( \text{John} \), \( \text{Mary} \), \( \text{Jane} \), …
- **Properties:** gender
  
  \( \text{Male} (x) \), \( \text{Female} (x) \)
- **Relations:** parenthood, brotherhood, marriage
  
  \( \text{Parent} (x, y) \), \( \text{Brother} (x, y) \), \( \text{Spouse} (x, y) \)
- **Functions:** mother-of (one for each person \( x \))
  
  \( \text{MotherOf} (x) \)

---

**Kinship domain in FOL**

**Relations between predicates and functions:** write down what we know about them; how relate to each other.

- Male and female are disjoint categories
  
  \( \forall x \text{ Male} (x) \iff \neg \text{Female} (x) \)
- Parent and child relations are inverse
  
  \( \forall x, y \text{ Parent} (x, y) \iff \text{Child} (y, x) \)
- A grandparent is a parent of parent
  
  \( \forall g, c \text{ Grandparent}(g, c) \iff \exists p \text{ Parent}(g, p) \land \text{Parent}(p, c) \)
- A sibling is another child of one’s parents
  
  \( \forall x, y \text{ Sibling} (x, y) \iff (x \neq y) \land \exists p \text{ Parent}(p, x) \land \text{Parent}(p, y) \)
- And so on ….
Knowledge engineering in FOL

1. Identify the problem/task you want to solve  
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
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 attributes: size, shape, color, cost of gates

3. **Decide on a vocabulary**
   - Alternatives:
     - Type($X_1$) = XOR
     - Type($X_1$, XOR)
     - XOR($X_1$)

4. **Encode general knowledge of the domain**
   - $\forall t_1,t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
   - $\forall t \text{ Signal}(t) = 1 \lor \text{Signal}(t) = 0$
   - $1 \neq 0$
   - $\forall t_1,t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$

   - $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 1$
   - $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 0$
   - $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
   - $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$
The electronic circuits domain

5. Encode the specific problem instance

Type($X_1$) = XOR
Type($X_2$) = XOR
Type($A_1$) = AND
Type($A_2$) = AND
Type($O_1$) = OR

Connected(Out(1,$X_1$),In(1,$X_2$))
Connected(In(1,$C_1$),In(1,$X_1$))
Connected(Out(1,$X_1$),In(2,$A_2$))
Connected(In(1,$C_1$),In(1,$A_1$))
Connected(Out(1,$A_2$),In(1,$O_1$))
Connected(In(2,$C_1$),In(2,$X_1$))

...
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?

\[ \exists i_1, i_2, i_3, o_1, o_2 \text{ Signal(In}(1, C_1)) = i_1 \land \text{Signal(In}(2, C_1)) = i_2 \land \text{Signal(In}(3, C_1)) = i_3 \land \text{Signal(Out}(1, C_1)) = o_1 \land \text{Signal(Out}(2, C_1)) = o_2 \]

7. **Debug the knowledge base**

May have omitted assertions like 1 ≠ 0

---

**Inference in First order logic**
Logical inference in FOL

Logical inference problem:
• Given a knowledge base $KB$ (a set of sentences) and a sentence $\alpha$, does the $KB$ semantically entail $\alpha$?

$$KB \models \alpha ?$$

In other words: In all interpretations in which sentences in the $KB$ are true, is also $\alpha$ true?

Logical inference problem in the first-order logic is undecidable !!!. No procedure that can decide the entailment for all possible input sentences in a finite number of steps.

Logical inference problem in the Propositional logic

Computational procedures that answer:

$$KB \models \alpha ?$$

Three approaches:
• Truth-table approach
• Inference rules
• Conversion to the inverse SAT problem
  – Resolution-refutation
Inference in FOL: Truth table

• Is the Truth-table approach a viable approach for the FOL?
  ?

Inference in FOL: Truth table approach

• Is the Truth-table approach a viable approach for the FOL?
  ?
  • **NO!**
  • Why?
  • It would require us to enumerate and list all possible interpretations $I$
  • $I =$ (assignments of symbols to objects, predicates to relations and functions to relational mappings)
  • Simply there are too many interpretations
Inference in FOL: Inference rules

- Is the Inference rule approach a viable approach for the FOL? 
- **Yes.**
- The inference rules represent sound inference patterns one can apply to sentences in the KB
- What is derived follows from the KB
- **Caveat:**
  - we need to add rules for handling quantifiers
Inference rules

- **Inference rules from the propositional logic**:
  - Modus ponens
    \[
    A \Rightarrow B, \quad A \\
    \frac{}{B}
    \]
  - Resolution
    \[
    A \lor B, \quad \neg B \lor C \\
    \frac{}{A \lor C}
    \]
  - and others: And-introduction, And-elimination, Or-introduction, Negation elimination

- **Additional inference rules** are needed for sentences with quantifiers and variables
  - Must involve variable substitutions

Variable substitutions

- Variables in the sentences can be substituted with terms.
  (terms = constants, variables, functions)

- **Substitution**:
  - Is a mapping from **variables to terms**
    \[
    \{x_1 / t_1, x_2 / t_2, \ldots\}
    \]
  - Application of the substitution to sentences
    \[
    \text{SUBST}(\{x / \text{Sam}, y / \text{Pam}\}, \text{Likes}(x, y)) = \text{Likes}(\text{Sam}, \text{Pam})
    \]
    \[
    \text{SUBST}(\{x / z, y / \text{fatherof (John)}\}, \text{Likes}(x, y)) = \text{Likes}(z, \text{fatherof (John)})
    \]
Inference rules for quantifiers

• **Universal elimination**
  \[ \forall x \, \phi(x) \quad \frac{\phi(a)}{} \quad a \text{ - is a constant symbol} \]
  – substitutes a variable with a **constant symbol**

• **Example:**
  \[ \forall x \, \text{Likes}(x, \text{IceCream}) \]
  \[ \text{Likes}(\text{Ben}, \text{IceCream}) \]

• **Existential elimination**
  \[ \exists x \, \phi(x) \quad \frac{\phi(a)}{} \quad \text{Substitutes a variable with a **constant symbol** that does not appear elsewhere in the KB} \]

• **Examples:**
  • \( \exists x \, \text{Kill}(x, \text{Victim}) \quad \text{Kill}(%03D{\text{Murderer}}, \text{Victim}) \)
    Special constant called **Skolem** constant
  • \( \exists x \, \text{Crown}(x) \land \text{OnHead}(x, \text{John}) \)
    \( \text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John}) \)
Reduction to propositional inference

Suppose the KB contains just the following:

\[ \forall x \text{King}(x) \land \text{Greedy}(x) \implies \text{Evil}(x) \]

\text{King}(\text{John})

\text{Greedy}(\text{John})

\text{Brother}(\text{Richard}, \text{John})

• Instantiating the universal sentence in all possible ways, we have:

\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \implies \text{Evil}(\text{John})

\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \implies \text{Evil}(\text{Richard})

\text{King}(\text{John})

\text{Greedy}(\text{John})

\text{Brother}(\text{Richard}, \text{John})

• The new KB is \textit{propositionalized}: proposition symbols are

\text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Evil}(\text{John}), \text{King}(\text{Richard}), \text{etc.}

Reduction contd.

• Every FOL KB can be \textit{propositionalized} so as to preserve entailment

• (A ground sentence is entailed by new KB iff entailed by original KB)

• \textbf{Idea of the inference:}
  – propositionalize KB and query, apply resolution, return result

• \textbf{Problem:} with function symbols, there are infinitely many ground terms,
  – e.g., \textit{Father}(\textit{Father}(\textit{Father}(\text{John})))
Reduction contd.

**Theorem:** Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to $\infty$ do
- create a propositional KB by instantiating with depth-$n$ terms
- see if $\alpha$ is entailed by this KB

**Problem:** works if $\alpha$ is entailed, loops if $\alpha$ is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is *semidecidable* (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)