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

  $V : \text{sentence} \times \text{interpretation} \rightarrow \{True, False\}$
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)
  
  \[
  \text{father-of}(\text{father-of}(\text{John}))
  \]

Sentences in FOL:
- **Atomic sentences**:
  - A **predicate symbol** applied to 0 or more terms
    
    **Examples**:
    
    \[
    \begin{align*}
    \text{Red(car12)}, \\
    \text{Sister(Amy, Jane);} \\
    \text{Manager(father-of(John));}
    \end{align*}
    \]
  
  - \( t_1 = t_2 \) **equivalence** of terms
    
    **Example**:
    
    \[
    \text{John} = \text{father-of}(\text{Peter})
    \]
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 $D$
  $I(John) = \text{John}$
• Predicate symbols to relations, properties on $D$
  $I(brother) = \{ \langle \text{John}, \text{Mary} \rangle; \langle \text{John}, \text{Susan} \rangle; \ldots \}$
• Function symbols to functional relations on $D$
  $I(father-of) = \{ \langle \text{John} \rangle \rightarrow \text{Mary}; \langle \text{Joseph} \rangle \rightarrow \text{Mary}; \ldots \}$
Semantics of sentences.

Meaning (evaluation) function:

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

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

\[
\begin{align*}
I(\text{John}) &= \begin{cases} 0 \end{cases} \\
I(\text{Paul}) &= \begin{cases} 1 \end{cases} \\
I(\text{brother}) &= \left\{ \left\langle \begin{cases} 0 \end{cases}, \begin{cases} 0 \end{cases} \right\rangle; \left\langle \begin{cases} 0 \end{cases}, \begin{cases} 1 \end{cases} \right\rangle; \ldots \right\} \\
V(\text{brother}(\text{John}, \text{Paul}), I) &= \text{True}
\end{align*}
\]

Semantics of sentences.

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

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

• **Quantifications**
  \[ V(\forall x \phi, I) = \text{True} \]
  Iff for all \( d \in D \) \( V(\phi, I[x/d]) = \text{True} \)
  \[ V(\exists x \phi, I) = \text{True} \]
  Iff there is a \( d \in D \), s.t. \( V(\phi, I[x/d]) = \text{True} \)
Note on the domain of discourse

• Can the domain of discourse be an empty set?
• Answer: No.
• Reason:
  – many equivalences in the logic would become false for the empty set and would have to be dealt with separately
• Example:
  \[ \exists x \ (\phi \lor \varphi(x)) \iff (\phi \lor \exists x \varphi(x)) \]

• Assume: \( \phi = \text{True} \) then:
  \[ \text{False} \iff \text{True} \]

Order of quantifiers

• Order of quantifiers of the same type does not matter
  \[ \forall x, y \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x) \]
  \[ \forall y, x \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x) \]

• Order of different quantifiers changes the meaning
  \[ \forall x \exists y \ \text{loves} \ (x, y) \]
  Everybody loves somebody
  \[ \exists y \forall x \ \text{loves} \ (x, y) \]
  There is someone who is loved by everyone
Connections between quantifiers

Everyone likes ice cream

\( \forall x \, \text{likes} (x, \text{IceCream} ) \)

Is it possible to convey the same meaning using an existential quantifier?

There is no one who does not like ice cream

\( \neg \exists x \, \neg \text{likes} (x, \text{IceCream} ) \)

A universal quantifier in the sentence can be expressed using an existential quantifier !!!

Connections between quantifiers

Someone likes ice cream

\( \exists x \, \text{likes} (x, \text{IceCream} ) \)

Is it possible to convey the same meaning using a universal quantifier?

Not everyone does not like ice cream

\( \neg \forall x \, \neg \text{likes} (x, \text{IceCream} ) \)

An existential quantifier in the sentence can be expressed using a universal quantifier !!!
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(X₁) = XOR
     Type(X₁, XOR)
     XOR(X₁)

4. Encode general knowledge of the domain
   - \( \forall t₁, t₂ \) Connected(t₁, t₂) \( \Rightarrow \) Signal(t₁) = Signal(t₂)
   - \( \forall t \) Signal(t) = 1 \( \lor \) Signal(t) = 0
   - 1 \( \neq \) 0
   - \( \forall t₁, t₂ \) Connected(t₁, t₂) \( \Rightarrow \) Connected(t₂, t₁)
   - \( \forall g \) Type(g) = OR \( \Rightarrow \) Signal(Out(1,g)) = 1 \( \Leftrightarrow \) \( \exists n \) Signal(In(n,g)) = 1
   - \( \forall g \) Type(g) = AND \( \Rightarrow \) Signal(Out(1,g)) = 0 \( \Leftrightarrow \) \( \exists n \) Signal(In(n,g)) = 0
   - \( \forall g \) Type(g) = XOR \( \Rightarrow \) Signal(Out(1,g)) = 1 \( \Leftrightarrow \) Signal(In(1,g)) \( \neq \) Signal(In(2,g))
   - \( \forall g \) Type(g) = NOT \( \Rightarrow \) Signal(Out(1,g)) \( \neq \) Signal(In(1,g))
The electronic circuits domain

5. Encode the specific problem instance

Type(X₁) = XOR
Type(X₂) = XOR
Type(A₁) = AND
Type(A₂) = AND
Type(O₁) = OR

Connected(Out(1,X₁),In(1,X₂))
Connected(In(1,C₁),In(1,X₁))
Connected(Out(1,X₁),In(2,A₂))
Connected(In(1,C₁),In(1,A₁))
Connected(Out(1,A₂),In(1,O₁))
Connected(In(2,C₁),In(2,X₁))

...
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 \( 1 \neq 0 \)

---

Representing knowledge in FOL

**Example:**

**Kinship domain**

- **Objects:** people
  
  \( John, Mary, Jane, \ldots \)

- **Properties:** gender
  
  \( Male(x), Female(x) \)

- **Relations:** parenthood, brotherhood, marriage
  
  \( Parent(x, y), \) \( Brother(x, y), \) \( Spouse(x, y) \)

- **Functions:** mother-of (one for each person \( x \))
  
  \( MotherOf(x) \)
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 ....
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 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
    \[
    \frac{A \Rightarrow B, \ A}{B}
    \]
  - Resolution
    \[
    \frac{A \lor B, \ \neg B \lor C}{A \lor C}
    \]
  - 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**
  \[ \forall x \phi(x) \]
  \[ \phi(a) \quad a \text{ - is a constant symbol} \]
  - substitutes a variable with a **constant symbol**

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

- **Existential elimination**
  \[ \exists x \phi(x) \]
  \[ \phi(a) \]
  - Substitutes a variable with a **constant symbol** that does not appear elsewhere in the KB

- **Examples:**
  - \[ \exists x \text{Kill}(x, \text{Victim}) \quad \rightarrow \quad \text{Kill(} \text{Murderer}, \text{Victim} \text{)} \]
    - Special constant called a **Skolem** constant
  - \[ \exists x \text{Crown}(x) \land \text{OnHead}(x, \text{John}) \]
    \[ \rightarrow \]
    \[ \text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John}) \]
Reduction to propositional inference

Suppose the KB contains just the following:

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

- King(John)
- Greedy(John)
- Brother(Richard,John)

• Instantiating the universal sentence in all possible ways, we have:
  - King(John) \land Greedy(John) \Rightarrow Evil(John)
  - King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)
  - King(John)
  - Greedy(John)
  - Brother(Richard,John)

• The new KB is propositionalized: proposition symbols are
  - King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction contd.

• Every FOL KB can be propositionalized so as to preserve entailment
• (A ground sentence is entailed by new KB iff it is entailed by the original KB)

• Idea of the inference:
  - propositionalize KB and query, apply resolution, return result

• Problem: with function symbols, there are infinitely many ground terms,
  - e.g., Father(Father(Father(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.)

---

**Problems with propositionalization**

- **Propositionalization** seems to generate lots of irrelevant sentences
- E.g., from:
  \[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \]
  King(John)
  \[ \forall y \ Greedy(y) \]
  Brother(Richard, John)
- It seems obvious that Evil(John), but propositionalization produces lots of facts such as Greedy(Richard) that are irrelevant
- With $p$ $k$-ary predicates and $n$ constants, there are $p \cdot n^k$ instantiations.