Formal Methods in Software Engineering

First-Order Logic — Spring 2025

Konstantin Chukharev

§1 Introduction to FOL

Motivation

Propositional logic (PL) is not powerful enough for many applications.

- PL cannot reason about *natural numbers* directly.
- PL cannor reason about *infinite* domains.
- PL cannot express *abstract* properties.
- PL cannot respresent the *internal structure* of propositions.
- PL lacks *quantifiers* for generalization.

First-order logic (FOL) *extends* propositional logic by adding *variables*, *predicates*, *functions*, and *quantifiers*, providing a structured way to reason about objects, their properties, and relationships.

Unlike propositional logic, which is limited to fixed truth values for statements, FOL allows complex expressions like "All humans are mortal" ($\forall x.\, \text{Human}(x) \rightarrow \text{Mortal}(x)$) or "There exists a solution to the problem" ($\exists x.\, \text{Solution}(x)$).

What is First-Order Logic?

Similar to PL, first-order logic is a formal system with a *syntax* and *semantics*.

First-order logic is an umbrella term for different *first-order languages*.

Syntax of a logic consists of *symbols* and *rules* for combining them into *well-formed formulas* (WFFs).

Symbols of a first-order language are divided into *logical symbols* and *non-logical parameters*.

Logical symbols:

- Parantheses: $(,)$
- Logical connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Variables: $x, y, z, \ldots$
- Quantifiers: $\forall$ and $\exists$

Parameters:

- Equiality: $=$
- Constants: e.g. $\bot, 0, \emptyset, \ldots$
- Predicates: e.g. $P(x), Q(x, y), x > y, A \subset B$, a $\prec$ ab
- Functions: e.g. $f(x), g(x, y), x + y, x \oplus y$

Note: For connectives, just $\wedge$ and $\neg$ are enough. The others can be expressed in terms of them.

Note: For quantifiers, just $\forall$ is enough, since $\exists x.\, \varphi$ can be expressed as $\neg \forall x.\, \neg \varphi$.

Predicates and Functions

Predicates are used to express properties or relations among objects.

Functions are similar to predicates but return a value, not necessarily a truth value.

Each predicate and function symbol has a fixed *arity* (number of arguments).

- Equality is a special predicate with arity 2.
- Constants can be seen as functions with arity 0.

First-Order Language

First-order language is specified by its *parameters*.

Propositional logic:
- Equiality: *no*
- Constants: *none*
- Predicates: A_1, A_2, ...
- Functions: *none*

Set theory:
- Equiality: *yes*
- Constants: $\emptyset$
- Predicates: $\in$
- Functions: *none*

Number theory:
- Equiality: *yes*
- Constants: 0
- Predicates: $<$
- Functions: S (successor), $+$, $\times$, exp

§2 Many-Sorted FOL

Syntax

The *syntax* of a logic consists of *symbols* and *rules* for combining them.

The *symbols* of a first-order language include:

1. Logical symbols: $(,)$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$
2. Infinite set of variables: x, y, z, ...
3. Signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$, where:
   - Σ^S is a set of *sorts* (also called *types*), e.g. `Bool`, Int, Real, `Set`.
   - Σ^F is a set of *function symbols*, e.g. $=$, $+$, $<$, ...

Signatures

Definition 1: Signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$ consists of:
- Σ^S is a set of *sorts* ~~(also called *types*)~~, e.g. `Bool`, `Int`, `Real`, `Set`
- Σ^F is a set of *function symbols*, e.g. $=, +, <$

Definition 2: Each *function symbol* $f \in \Sigma^F$ is associated with an *arity* n (number of arguments) and a *rank*, $(n+1)$-tuple of sorts: $\text{rank}(f) = \langle \sigma_1, \sigma_2, ..., \sigma_{n+1} \rangle$. Intuitively, f denotes a function that takes n values of sorts $\sigma_1, ..., \sigma_n$ and returns an output of sort σ_{n+1}.
- Functions of arity 0 are called *constants*, which are said to have sort σ if $\text{rank}(f) = \langle \sigma \rangle$.
- Functions that *return* sort `Bool` are called *predicates*.

For every signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$, we assume that:
- Σ^S includes a distinguished sort `Bool`.
- Σ^F contains distinguished constants $\top$ and $\bot$ of sort `Bool`, and distinguished predicate symbol $\doteq$ with $\text{rank}(\doteq) = \langle \sigma, \sigma, \texttt{Bool} \rangle$ for every sort $\sigma \in \Sigma^S$.

Equality

TODO: axioms of equality

- reflexivity
- substitution for functions
- substitution for formulas

First-Order Languages

A first-order language is defined w.r.t. a signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$.

Number Theory:

- $\Sigma^S = \{\mathtt{Nat}\} \cup \{\mathtt{Bool}\}$
- $\Sigma^F = \{0, S, <, +, \times\} \cup \{\top, \bot, \doteq_{\mathtt{Bool}}, \doteq_{\mathtt{Nat}}\}$
  - $\mathrm{rank}(0) = \langle \mathtt{Nat} \rangle$
  - $\mathrm{rank}(S) = \langle \mathtt{Nat}, \mathtt{Nat} \rangle$
  - $\mathrm{rank}(<) = \langle \mathtt{Nat}, \mathtt{Nat}, \mathtt{Bool} \rangle$
  - $\mathrm{rank}(+) = \mathrm{rank}(\times) = \langle \mathtt{Nat}, \mathtt{Nat}, \mathtt{Nat} \rangle$

Set Theory:

- $\Sigma^S = \{\mathtt{Set}\} \cup \{\mathtt{Bool}\}$
- $\Sigma^F = \{\emptyset, \in, \cup, \cap\} \cup \{\top, \bot, \doteq_{\mathtt{Bool}}, \doteq_{\mathtt{Set}}\}$
  - $\mathrm{rank}(\emptyset) = \langle \mathtt{Set} \rangle$
  - $\mathrm{rank}(\in) = \langle \mathtt{Set}, \mathtt{Set}, \mathtt{Bool} \rangle$
  - $\mathrm{rank}(\cup) = \mathrm{rank}(\cap) = \langle \mathtt{Set}, \mathtt{Set}, \mathtt{Set} \rangle$

Propositional Logic:

- $\Sigma^S = \{\mathtt{Bool}\}$
- $\Sigma^F = \{\neg, \wedge, \vee, ..., p_1, p_2, ...\} \cup \{\top, \bot, \doteq_{\mathtt{Bool}}\}$
  - $\mathrm{rank}(p_i) = \langle \mathtt{Bool} \rangle$
  - $\mathrm{rank}(\neg) = \langle \mathtt{Bool}, \mathtt{Bool} \rangle$
  - $\mathrm{rank}(\wedge) = \mathrm{rank}(\vee) = \langle \mathtt{Bool}, \mathtt{Bool}, \mathtt{Bool} \rangle$

Arrays Theory:

- $\Sigma^S = \left\{\mathtt{Array}_{\langle \mathtt{X},\mathtt{Y} \rangle}\right\} \cup \{\mathtt{Bool}\}$
  - $\mathtt{X}$ is a sort of *indices*.
  - $\mathtt{Y}$ is a sort of *values*.
- $\Sigma^F = \{\mathrm{read}, \mathrm{write}\} \cup \{\top, \bot, \doteq_{\mathtt{Bool}}, \doteq_{\mathtt{Array}}\}$
  - $\mathrm{rank}(\mathrm{read}) = \langle \mathtt{Array}_{\langle \mathtt{X},\mathtt{Y} \rangle}, \mathtt{X}, \mathtt{Y} \rangle$
  - $\mathrm{rank}(\mathrm{write}) = \langle \mathtt{Array}_{\langle \mathtt{X},\mathtt{Y} \rangle}, \mathtt{X}, \mathtt{Y}, \mathtt{Array}_{\langle \mathtt{X},\mathtt{Y} \rangle} \rangle$

Expressions

Definition 3: An *expression* is a finite sequence of symbols.

Examples:

- $\forall x_1((< 0\ x_1) \to \neg\forall x_2(< x_1\ x_2))$
- $x_1 < \forall x_2))$
- $x_1 < x_2 \to \forall(x : \mathrm{Nat})x > 0$

Note: Most expressions are **not** *well-formed*.

Terms (simple)

Definition 4 (Variables): A set X of Σ-variables, or simply *variables*, is a countable set of variable names, each associated with a sort from Σ^S.

Based on variables, we can build *terms*. Intuitevely, terms are expressions that evaluate to values.

Definition 5 (Terms): The Σ-terms over X, or simply *terms*, are defined inductively:

- Each variable x in X is a term of sort σ.
- If $c \in \Sigma^F$ is a constant symbol of sort σ, then c is a term of sort σ.
- If $t_1, ..., t_n$ are terms of sorts $\sigma_1, ..., \sigma_n$, and f is a function symbol with $\text{rank}(f) = \langle \sigma_1, ..., \sigma_n, \sigma \rangle$, then $(f\ t_1 \cdots t_n)$ is a term of sort σ.
- *(Nothing else is a term.)*

Formulas (simple)

Based on terms, we can build *atoms*. Intuitevely, atoms are expressions that evaluate Boolean values.

Definition 6 (Atoms): The Σ-atoms over X, or simply *atoms*, are terms of the form $(p\ t_1 \cdots t_n)$, where $t_1, ..., t_n$ are terms of sorts $\sigma_1, ..., \sigma_n$, and p is a predicate symbol with $\mathrm{rank}(p) = \langle \sigma_1, ..., \sigma_n, \texttt{Bool} \rangle$.

In addition to sorted Σ-variables X, also consider *propositional variables* $\mathcal{B}$.

Definition 7 (Formulas): The Σ-formulas over X and $\mathcal{B}$, or simply *formulas*, are defined inductively:

- Each propositional variable p in $\mathcal{B}$ is a formula.
- Each Σ-atom over X is a formula.
- If α and β are formulas, then so are $\neg\alpha$, $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\alpha \to \beta)$, and $(\alpha \leftrightarrow \beta)$.
- For each variable $x \in X$ and sort $\sigma \in \Sigma^S$, if α is a formula, then so are $\forall x : \sigma.\, \alpha$ and $\exists x : \sigma.\, \alpha$.

Terms

A *term* is a *well-formed* S-expression built from function symbols, variables, and parentheses.

Definition 8 (Term): Let $\mathcal{B}$ be the set of all variables and all constant symbols in some signature Σ.

For each function symbol f in Σ^F of arity n, define *term-building operation* $\mathcal{T}_f$:

$$\mathcal{T}_f(\varepsilon_1, ..., \varepsilon_n) := (f\ \varepsilon_1 \cdots \varepsilon_1)$$

Well-formed terms are expressions generated from $\mathcal{B}$ by $\mathcal{T} = \{\mathcal{T}_f \mid f \in \Sigma^F\}$.

Examples:

✓ $(+\ x_2\ (S\ 0))$
✓ $(S\ (S\ (S\ (S\ 0))))$
✗ $(S\ (0\ 0))$

✗ $(x_2 + 0)$
✗ $(S\ 0\ 0)$
✓ $(S\ (<\ 0\ 0))$

✓ $(+\ x_2\ \bot)$
✓ $(S\ \bot)$
✓ $(\doteq\ 0\ \bot)$

✗ $(\text{select}\ a)$
✓ $(\text{select}\ a\ i)$
✓ $(\text{select}\ (\text{store}\ a\ i\ x)\ j)$

Well-sortedness

Note: Not all well-formed terms are are meaningful. For this, we need to take into account *sorts*.

Definition 9: *Sort system* is a proof system over sequents of the form $\Gamma \vdash t : \sigma$.

- $\Gamma = x_1 : \sigma_1, ..., x_n : \sigma_n$ is a *sort context*, a set of sorted variables.
- t is a well-formed term.
- σ is a sort from Σ^S.

$$\textsc{Var}\ \frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \textsc{Const}\ \frac{c \in \Sigma^F \quad \mathrm{rank}(c) = \langle \sigma \rangle}{\Gamma \vdash c : \sigma}$$

$$\textsc{Fun}\ \frac{f \in \Sigma^F \quad \mathrm{rank}(f) = \langle \sigma_1, ..., \sigma_n, \sigma \rangle \quad \Gamma \vdash t_1 : \sigma_1 \quad \cdots \quad \Gamma \vdash t_n : \sigma_n}{\Gamma \vdash (f\ t_1 \cdots t_n) : \sigma}$$

Definition 10 (Σ-term): A term t is *well-sorted* w.r.t. Σ and *has sort* σ in a sort context Γ if $\Gamma \vdash t : \sigma$ is derivable in the sort system. Term t is called Σ-term.

Examples of Well-sorted Terms

Let $\Sigma^S = \{\texttt{Nat}\} \cup \{\texttt{Bool}\}$ and $\Sigma^F = \{0, S, <, +, \times, \doteq_{\texttt{Nat}}\} \cup \{\top, \bot, \doteq_{\texttt{Bool}}\}$.

- $\text{rank}(0) = \langle \texttt{Nat} \rangle$
- $\text{rank}(S) = \langle \texttt{Nat}, \texttt{Nat} \rangle$
- $\text{rank}(<) = \text{rank}(\doteq_{\texttt{Nat}}) = \langle \texttt{Nat}, \texttt{Nat}, \texttt{Bool} \rangle$
- $\text{rank}(+) = \text{rank}(\times) = \langle \texttt{Nat}, \texttt{Nat}, \texttt{Nat} \rangle$

Are these well-formed terms also well-sorted in the context $\Gamma = \{x_1 : \texttt{Bool}, x_2 : \texttt{Nat}, x_3 : \texttt{Nat}\}$?

1. $(+\ 0\ x_2)$ ✓
2. $(+\ (+\ 0\ x_1)\ x_2)$ ✗
3. $(S\ (+\ 0\ x_5))$ ✓
4. $(<\ (S\ x_3)\ (+\ (S\ 0)\ x_1))$ ✓
5. $(\doteq_{\texttt{Nat}}\ (S\ x_3)\ (+\ (S\ 0)\ x_1))$ ✓

Formulas

Definition 11 (Σ-atom): Given a signature Σ, an *atomic Σ-formula*, or simply *atom*, is a Σ-term of sort `Bool` under *some* sort context Γ.

Definition 12 (Formula): *Well-formed formulas* are expressions generated from atoms by the *formula-building operations*, denoted $\mathcal{F} = \left\{\mathcal{F}_\vee, \mathcal{F}_\wedge, \mathcal{F}_\neg, \mathcal{F}_\to, \mathcal{F}_\leftrightarrow, \mathcal{E}_{x,\sigma}, \mathcal{A}_{x,\sigma}\right\}$.

- $\mathcal{F}_\vee(\alpha, \beta) := (\alpha \vee \beta)$
- $\mathcal{F}_\wedge(\alpha, \beta) := (\alpha \wedge \beta)$
- $\mathcal{F}_\neg(\alpha) := (\neg\alpha)$
- $\mathcal{F}_\to(\alpha, \beta) := (\alpha \to \beta)$
- $\mathcal{F}_\leftrightarrow(\alpha, \beta) := (\alpha \leftrightarrow \beta)$
- $\mathcal{E}_{x,\sigma}(\alpha) := (\exists x : \sigma.\, \alpha)$ for each variable x and sort σ in Σ^S
- $\mathcal{A}_{x,\sigma}(\alpha) := (\forall x : \sigma.\, \alpha)$ for each variable x and sort σ in Σ^S

Examples of Formulas

Let $\Sigma^S = \{\texttt{Nat}\}$, $\Sigma^F = \{0, S, <, +, \times, \doteq_{\texttt{Nat}}\}$, and let x_i be variables.

Which of the following formulas are well-formed?

1. $(\doteq_{\texttt{Nat}}\ 0\ (S\ 0))$ ✓
2. $(<\ (S\ x_3)\ (+\ (S\ 0)\ x_1))$ ✓
3. $(\doteq_{\texttt{Nat}}\ (+\ x_1\ 0)\ x_2)$ ✓
4. $(\doteq_{\texttt{Nat}}\ (+\ x_1\ 0)\ x_2) \to \bot$ ✓
5. $(+\ 0\ x_3) \wedge (<\ 0\ (S\ 0))$ ✗
6. $\forall x_3 : \texttt{Nat}.\ (+\ (+\ 0\ x_3)\ x_2)$ ✗
7. $\forall x_3 : \texttt{Bool}.\ (\doteq_{\texttt{Nat}}\ (+\ 0\ x_3)\ x_2)$ ✓ *(Note: not well-sorted)*
8. $\neg\exists x_0 : \texttt{Nat}.\ (<\ 0\ x_0\ (S\ 0))$ ✗

Well-sorted Formulas

We *extend* the sort system for terms with rules for the *logical connectives* and *quantifiers*.

$$\textsc{Bconst}\ \frac{c \in \{\top, \bot\}}{\Gamma \vdash c : \texttt{Bool}} \qquad \textsc{Not}\ \frac{\Gamma \vdash \alpha : \texttt{Bool}}{\Gamma \vdash (\neg\alpha) : \texttt{Bool}}$$

$$\textsc{Conn}\ \frac{\Gamma \vdash \alpha : \texttt{Bool} \quad \Gamma \vdash \beta : \texttt{Bool} \quad \bowtie \in \{\wedge, \vee, \to, \leftrightarrow\}}{\Gamma \vdash (\alpha \bowtie \beta) : \texttt{Bool}}$$

$$\textsc{Quant}\ \frac{\Gamma[x : \sigma] \vdash \alpha : \texttt{Bool} \quad \sigma \in \Sigma^S \quad \mathrm{Q} \in \{\forall, \exists\}}{\Gamma \vdash (\mathrm{Q}\, x : \sigma.\, \alpha) : \texttt{Bool}}$$

Here, $\Gamma[x : \sigma] = \Gamma \cup \{x : \sigma\}$.

Definition 13 (Σ-formula): A formula φ is a *well-sorted* w.r.t. Σ in a sort context Γ if $\Gamma \vdash \varphi : \texttt{Bool}$ is derivable in the extended sort system. Formula φ is called Σ-formula.

Free and Bound Variables

A variable x may occur *free* or *bound* in a Σ-formula.

Definition 14: The set $\mathcal{FV}$ of *free variables* of a Σ-formula α is defined as follows:

$$\mathcal{FV}(\alpha) := \begin{cases} \{x \mid x \text{ is a var in } \alpha\} & \text{if } \alpha \text{ is atomic} \\ \mathcal{FV}(\beta) & \text{if } \alpha \equiv \neg\beta \\ \mathcal{FV}(\beta) \cup \mathcal{FV}(\gamma) & \text{if } \alpha \equiv (\alpha \bowtie \beta) \text{ with } \bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\ \mathcal{FV}(\beta) \setminus \{v\} & \text{if } \alpha \equiv Qv : \sigma.\, \beta \text{ with } Q \in \{\forall, \exists\} \end{cases}$$

Example: Let x, y, and z be variables.

- $\mathcal{FV}(x) = \{x\}$ (if x has sort `Bool`)
- $\mathcal{FV}(x < S(0) + y) = \{x, y\}$
- $\mathcal{FV}((x < S(0) + y) \wedge (x \doteq z)) = \mathcal{FV}(x < S(0) + y) \cup \mathcal{FV}(x \doteq z) = \{x, y\} \cup \{x, z\} = \{x, z, y\}$
- $\mathcal{FV}(\forall x : \texttt{Nat}.\, x < S(0) + y) = \mathcal{FV}(x < S(0) + y) \setminus \{x\} = \{x, y\} \setminus \{x\} = \{y\}$

Scope

- TODO: scope of variables
- TODO: free/bound variable
- TODO: open/closed formula
- TODO: universal closure
- TODO: existential closure
- TODO: variable can be simultaneously free and bound

FOL Semantics

Recall: The *syntax* of a first-order language is defined w.r.t. a signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$, where:

- Σ^S is a set of *sorts*.
- Σ^F is a set of *function symbols*.

In PL, the truth of a formula depends on the meaning of its variables.

In FOL, the truth of a Σ-formula depends on:

1. The meaning of each sort $\sigma \in \Sigma^S$ in the formula.
2. The meaning of each function symbol $f \in \Sigma^F$ in the formula.
3. The meaning of each free variable x in the formula.

Semantics

Let α be a Σ-formula and let Γ be a sort context that includes all free variables of α.

The truth of α is determined by *interpretations* $\mathcal{I}$ of Σ and Γ consisting of:

- An interpretation $\sigma^{\mathcal{I}}$ of each $\sigma \in \Sigma^S$, as a non-empty set, the *domain* of σ.
- An interpretation $f^{\mathcal{I}}$ of each $f \in \Sigma^F$ of rank $\langle \sigma_1, ..., \sigma_n, \sigma_{n+1} \rangle$, as an n-ary total function from $\sigma_1^{\mathcal{I}} \times \cdots \times \sigma_n^{\mathcal{I}}$ to $\sigma_{n+1}^{\mathcal{I}}$.
- An interpretation $x^{\mathcal{I}}$ of each $x : \sigma \in \Gamma$, as an element of $\sigma^{\mathcal{I}}$.

Note: We consider only interpretations $\mathcal{I}$ such that

- $\texttt{Bool}^{\mathcal{I}} = \{\texttt{true}, \texttt{false}\}, \quad \bot^{\mathcal{I}} = \texttt{false}, \quad \top^{\mathcal{I}} = \texttt{true}$,
- for all $\sigma \in \Sigma^S$, $\doteq_\sigma^{\mathcal{I}}$ maps its two arguments to `true` iff they are identical.

Semantics: Example

Consider a signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$ for a fragment of a set theory with non-set elements (ur-elements):

- $\Sigma^S = \{\texttt{Elem}, \texttt{Set}\}$,
- $\Sigma^F = \{\emptyset, \varepsilon\}$ with $\mathrm{rank}(\emptyset) = \langle \texttt{Set} \rangle$, $\mathrm{rank}(\varepsilon) = \langle \texttt{Elem}, \texttt{Set}, \texttt{Bool} \rangle$,
- $\Gamma = \{e_i : \texttt{Elem} \mid i \geq 0\} \cup \{s_i : \texttt{Set} \mid i \geq 0\}$

A possible *interpretation* $\mathcal{J}$ of Σ and Γ:

- $\texttt{Elem}^{\mathcal{J}} = \mathbb{N}$, the natural numbers.
- $\texttt{Set}^{\mathcal{J}} = 2^{\mathbb{N}}$, all sets of natural numbers.
- $\emptyset^{\mathcal{J}} = \emptyset$, the empty set.
- For all $n \in \mathbb{N}$ and $S \subseteq \mathbb{N}$, $\varepsilon^{\mathcal{J}}(n, S) \leftrightarrow n \in S$.
- For $i \geq 0$, $e_i^{\mathcal{J}} = i$ and $s_i^{\mathcal{J}} = [0; i] = \{0, 1, ..., i\}$.

Another *interpretation* $\mathcal{J}$ of Σ and Γ:

- $\texttt{Elem}^{\mathcal{J}} = \texttt{Set}^{\mathcal{J}} = \mathbb{N}$.
- $\emptyset^{\mathcal{J}} = 0$.
- For all $m, n \in \mathbb{N}$, $\varepsilon^{\mathcal{J}}(m, n) \leftrightarrow m \mid n$.
- For $i \geq 0$, $e_i^{\mathcal{J}} = i$ and $s_i^{\mathcal{J}} = 2$.

There is a *infinity* of interpretations of Σ, Γ.

Term and Formula Semantics

First, extend $\mathcal{I}$ to an interpretation $\overline{\mathcal{I}}$ for *well-sorted* Σ*-terms* by structural induction:

$$t^{\overline{\mathcal{I}}} = \begin{cases} t^{\mathcal{I}} & \text{if } t \text{ is a constant or a variable} \\ f^{\mathcal{I}}\left(t_1^{\overline{\mathcal{I}}}, ..., t_n^{\overline{\mathcal{I}}}\right) & \text{if } t \text{ is a term } (f\ t_1\ ...\ t_n) \end{cases}$$

Example: Let $\Sigma^S = \{\texttt{Person}\}$, $\Sigma^F = \{\text{pa}, \text{ma}, \text{sp}\}$, $\Gamma = \{x : \texttt{Person}, y : \texttt{Person}\}$, $\text{rank}(\text{pa}) = \text{rank}(\text{ma}) = \langle \texttt{Person}, \texttt{Person} \rangle$, $\text{rank}(\text{sp}) = \langle \texttt{Person}, \texttt{Person}, \texttt{Bool} \rangle$.

Let $\mathcal{I}$ be an interpretation of Σ and Γ such that:

- $\text{ma}^{\mathcal{I}} = \{\text{Jim} \mapsto \text{Jill}, \text{Joe} \mapsto \text{Jen}, ...\}$
- $\text{pa}^{\mathcal{I}} = \{\text{Jim} \mapsto \text{Joe}, \text{Jill} \mapsto \text{Jay}, ...\}$
- $\text{sp}^{\mathcal{I}} = \{(\text{Jill}, \text{Joe}) \mapsto \texttt{true}, (\text{Joe}, \text{Jill}) \mapsto \texttt{true}, (\text{Jill}, \text{Jill}) \mapsto \texttt{false}, ...\}$
- $x^{\mathcal{I}} = \text{Jim}$ and $y^{\mathcal{I}} = \text{Joe}$

Then:

Term and Formula Semantics [2]

- $(\text{pa}\ (\text{ma}\ x))^{\overline{\mathcal{J}}} = \text{pa}^{\mathcal{J}}((\text{ma}\ x)^{\overline{\mathcal{J}}}) = \text{pa}^{\mathcal{J}}(\text{ma}^{\mathcal{J}}(x^{\overline{\mathcal{J}}})) = \text{pa}^{\mathcal{J}}(x^{\mathcal{J}})$
  $= \text{pa}^{\mathcal{J}}(\text{ma}^{\mathcal{J}}(\text{Jim})) = \text{pa}^{\mathcal{J}}(\text{Jill}) = \text{Jay}$
- $(\text{sp}\ (\text{ma}\ x)\ y)^{\overline{\mathcal{J}}} = \text{sp}^{\mathcal{J}}((\text{ma}\ x)^{\overline{\mathcal{J}}}, y^{\overline{\mathcal{J}}}) = \text{sp}^{\mathcal{J}}(\text{ma}^{\mathcal{J}}(x^{\overline{\mathcal{J}}}), y^{\mathcal{J}}) = \text{sp}^{\mathcal{J}}(\text{ma}^{\mathcal{J}}(x^{\mathcal{J}}), y^{\mathcal{J}})$
  $= \text{sp}^{\mathcal{J}}(\text{ma}^{\mathcal{J}}(\text{Jim}), \text{Joe}) = \text{sp}^{\mathcal{J}}(\text{Jill}, \text{Joe}) = \texttt{true}$

Further extend $\overline{\mathcal{J}}$ to *well-sorted non-atomic* Σ*-formulas* by structural induction:

- $(\neg\alpha)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}} = \texttt{false}$
- $(\alpha \wedge \beta)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}} = \texttt{true}$ and $\beta^{\overline{\mathcal{J}}} = \texttt{true}$
- $(\alpha \vee \beta)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}} = \texttt{true}$ or $\beta^{\overline{\mathcal{J}}} = \texttt{true}$
- $(\alpha \to \beta)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}} = \texttt{false}$ or $\beta^{\overline{\mathcal{J}}} = \texttt{true}$
- $(\alpha \leftrightarrow \beta)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}} = \beta^{\overline{\mathcal{J}}}$
- $(\exists x : \sigma.\, \alpha)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}[x \mapsto c]} = \texttt{true}$ for some $c \in \sigma^{\mathcal{J}}$
- $(\forall x : \sigma.\, \alpha)^{\overline{\mathcal{J}}} = \texttt{true}$ iff $\alpha^{\overline{\mathcal{J}}[x \mapsto c]} = \texttt{true}$ for all $c \in \sigma^{\mathcal{J}}$

Here, $\overline{\mathcal{J}}[x \mapsto c]$ denotes the interpretation that maps x to c and is otherwise identical to $\overline{\mathcal{J}}$.

Satisfiability, Entailment, Validity

We write $\mathcal{I} \vDash \alpha$ to denote "$\mathcal{I}$ *satisfies* α" and mean $\alpha^{\overline{\mathcal{I}}} = \texttt{true}$.

We write $\mathcal{I} \nvDash \alpha$ to denote "$\mathcal{I}$ *falsifies* α" and mean $\alpha^{\overline{\mathcal{I}}} = \texttt{false}$.

Let Φ be a set of Σ-formulas. We write $\mathcal{I} \vDash \Phi$ to mean that $\mathcal{I} \vDash \alpha$ for every $\alpha \in \Phi$.

If Φ is a set of Σ-formulas and α is a Σ-formula, then Φ *entails* or *logically implies* α, denoted $\Phi \vDash \alpha$, if $\mathcal{I} \vDash \alpha$ for every interpretation $\mathcal{I}$ of Σ such that $\mathcal{I} \vDash \Phi$.

We write $\alpha \vDash \beta$ as an abbreviation for $\{\alpha\} \vDash \beta$.

α and β are *logically equivalent*, denoted $\alpha \equiv \beta$, iff $\alpha \vDash \beta$ and $\beta \vDash \alpha$.

A Σ-formula α is *valid*, denoted $\vDash \alpha$, if $\emptyset \vDash \alpha$ iff $\mathcal{I} \vDash \alpha$ for every interpretation $\mathcal{I}$ of Σ.

Example: Let $\Sigma^S = \{\texttt{A}\}$, $\Sigma^F = \{p, q\}$, $\text{rank}(p) = \langle \texttt{A}, \texttt{Bool} \rangle$, $\text{rank}(q) = \langle \texttt{A}, \texttt{A}, \texttt{Bool} \rangle$, and all variables v_i have sort $\texttt{A}$. Do the following entailments hold?

1. $\forall v_1 . p(v_1) \vDash p(v_2)$ ✓
2. $p(v_1) \vDash \forall v_1 . p(v_1)$ ✗

Satisfiability, Entailment, Validity [2]

3. $\forall p(v_1) \vDash \exists v_2.\, p(v_2)$ ✓
4. $\exists v_2\, \forall v_1.\, q(v_1, v_2) \vDash \forall v_1\, \exists v_2.\, q(v_1, v_2)$ ✓
5. $\forall v_1\, \exists v_2.\, q(v_1, v_2) \vDash \exists v_2\, \forall v_1.\, q(v_1, v_2)$ ✗
6. $\vDash \exists v_1.\, (p(v_1) \to \forall v_2.\, p(v_2))$ ✓

Exercise

Let α be a Σ-formula and let Γ be a sort context that includes all free variables of α.

Consider the signature where $\Sigma^S = \{\sigma\}$, $\Sigma^F = \{Q, \doteq_\sigma\}$, $\Gamma = \{x : \sigma, y : \sigma\}$, $\mathrm{rank}(Q) = \langle \sigma, \sigma, \texttt{Bool} \rangle$.

For each of the following Σ-formulas, describe an interpretation that satisfies it.

1. $\forall x : \sigma . \forall y : \sigma . x \doteq y$
2. $\forall x : \sigma . \forall y : \sigma . Q(x, y)$
3. $\forall x : \sigma . \exists y : \sigma . Q(x, y)$

From English to FOL

1. There is a natural number that is smaller than any *other* natural number.
   $\exists x : \texttt{Nat}.\, \forall y : \texttt{Nat}.\, (x \doteq y) \vee (x < y)$
2. For every natural number there is a greater one.
   $\forall x : \texttt{Nat}.\, \exists y : \texttt{Nat}.\, (x < y)$
3. Two natural numbers are equal only if their respective successors are equal.
   $\forall x : \texttt{Nat}.\, \forall y : \texttt{Nat}.\, (x \doteq y) \to (S(x) \doteq S(y))$
4. Two natural numbers are equal if their respective successors are equal.
   $\forall x : \texttt{Nat}.\, \forall y : \texttt{Nat}.\, (S(x) \doteq S(y)) \to (x \doteq y)$
5. No two distinct natural number have the same successor.
   $\forall x : \texttt{Nat}.\, \forall y : \texttt{Nat}.\, \neg(x \doteq y) \to \neg(S(x) \doteq S(y))$
6. There are at least two natural number smaller than 3.
   $\exists x : \texttt{Nat}.\, \exists y : \texttt{Nat}.\, \neg(x \doteq y) \wedge (x < S(S(S(0)))) \wedge (y < S(S(S(0))))$
7. There is no largest natural number.
   $\neg\exists x : \texttt{Nat}.\, \forall y : \texttt{Nat}.\, (y \doteq x) \vee (y < x)$

From English to FOL [2]

1. Everyone has a father and a mother.
   $\forall x : \texttt{Person}.\, \exists y : \texttt{Person}.\, \exists z : \texttt{Person}.\, (y \doteq \mathrm{pa}(x)) \wedge (z \doteq \mathrm{ma}(x))$
2. The marriage relation is symmetric.
   $\forall x : \texttt{Person}.\, \forall y : \texttt{Person}.\, \mathrm{sp}(x, y) \rightarrow \mathrm{sp}(y, x)$
3. No one can be married to themselves.
   $\forall x : \texttt{Person}.\, \neg\, \mathrm{sp}(x, x)$
4. Not all people are married.
   $\neg \forall x : \texttt{Person}.\, \exists y : \texttt{Person}.\, \mathrm{sp}(x, y)$
5. Some people have a farther and a mother who are not married to each other.
   $\exists x : \texttt{Person}.\, \neg\, \mathrm{sp}(\mathrm{ma}(x), \mathrm{pa}(x))$
6. You cannot marry more than one person.
   $\forall x : \texttt{Person}.\, \forall y : \texttt{Person}.\, \forall z : \texttt{Person}.\, (\mathrm{sp}(x, y) \wedge \mathrm{sp}(x, z)) \rightarrow (y \doteq z)$
7. Some people are not mothers.
   $\exists x : \texttt{Person}.\, \forall y : \texttt{Person}.\, \neg(x \doteq \mathrm{ma}(y))$

From English to FOL [3]

8. Nobody can be both a farther and a mother.
   $\forall x : \texttt{Person}. \neg\exists y : \texttt{Person}. \neg\exists z : \texttt{Person}. (x \doteq \mathrm{pa}(y)) \wedge (x \doteq \mathrm{ma}(z))$
9. Nobody can be their own or farther's farther.
   $\forall x : \texttt{Person}. \neg((x \doteq \mathrm{pa}(x)) \vee (x \doteq \mathrm{pa}(\mathrm{pa}(x))))$
10. Some people do not have children.
    $\exists x : \texttt{Person}. \forall y : \texttt{Person}. \neg(y \doteq \mathrm{pa}(x)) \wedge \neg(y \doteq \mathrm{ma}(y))$

Invariance of Term Values

Consider a signature Σ, a sort context Γ, and two interpretations $\mathcal{I}$ and $\mathcal{J}$ that agree on the sorts and symbols of Σ.

Theorem 1: If $\mathcal{I}$ and $\mathcal{J}$ also agree on the variables of a Σ-term t, then $t^{\overline{\mathcal{I}}} = t^{\overline{\mathcal{J}}}$.

Proof: By structural induction on t.

- If t is a variable or a constant, then $t^{\overline{\mathcal{I}}} = t^{\mathcal{I}}$ and $t^{\mathcal{J}} = t^{\overline{\mathcal{J}}}$. Since $t^{\mathcal{I}} = t^{\mathcal{J}}$ by assumption, we have $t^{\overline{\mathcal{I}}} = t^{\overline{\mathcal{J}}}$.
- If t is a term $(f\ t_1 \cdots t_n)$ with $n > 1$, then $f^{\mathcal{I}} = f^{\mathcal{J}}$ by assumption and $t_i^{\overline{\mathcal{I}}} = t_i^{\overline{\mathcal{J}}}$ for $j \geq 1$ by induction hypothesis. It follows, $t^{\overline{\mathcal{I}}} = f^{\mathcal{I}}(t_1^{\overline{\mathcal{I}}}, ..., t_n^{\overline{\mathcal{I}}}) = f^{\mathcal{J}}(t_1^{\overline{\mathcal{J}}}, ..., t_n^{\overline{\mathcal{J}}}) = t^{\overline{\mathcal{J}}}$.

□

Invariance of Truth Values

Theorem 2: If $\mathcal{I}$ and $\mathcal{J}$ also agree on the *free* variables of a Σ-formula α, then $\alpha^{\overline{\mathcal{I}}} = \alpha^{\overline{\mathcal{J}}}$.

Proof: By induction on α.

- If α is an atomic formula, the result follows from the previous lemma, since α is a term and all of its variables are free in it.
- If α is $\neg\beta$ or $\alpha_1 \bowtie \alpha_2$ with $\bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, the result follows from the induction hypothesis.
- If α is $\mathrm{Q}\, x : \sigma.\ \beta$ with $\mathrm{Q} \in \{\forall, \exists\}$, then $\mathcal{FV}(\beta) = \mathcal{FV}(\alpha) \cup \{x\}$. For any c in $\sigma^{\mathcal{I}}$, $\mathcal{I}[x \mapsto c]$ and $\mathcal{J}[x \mapsto c]$ agree on x by construction and on $\mathcal{FV}(\alpha)$ by assumption. The result follows from the induction hypothesis and the semantics of $\forall$ and $\exists$.

□

Corollary 2.1: The truth value of Σ-*sentences* is independent from how the variables are interpreted.

Deduction Theorem of FOL

Theorem 3: For all Σ-formulas α and β, we have $\alpha \vDash \beta$ iff $\alpha \to \beta$ is valid.

Proof ($\Rightarrow$): If $\alpha \vDash \beta$ then $\alpha \to \beta$ is valid.

Let $\mathcal{I}$ be an interpretation and let $\gamma := \alpha \to \beta$.

- If $\mathcal{I}$ falsifies α, then it trivially satisfies γ.
- If $\mathcal{I}$ satisfies α, then, since $\alpha \vDash \beta$, it must also satisfy β. Hence, it satisfies γ.

In both cases, $\mathcal{I}$ satisfies γ, thus $\mathcal{I} \vDash \alpha \to \beta$ for every interpretation $\mathcal{I}$. □

Proof ($\Leftarrow$): If $\alpha \to \beta$ is valid then $\alpha \vDash \beta$.

Let $\mathcal{I}$ be an interpretation that satisfies α. If $\mathcal{I}$ falsifies β, then it must also falsify $\alpha \to \beta$, contradicting the assumption that $\alpha \to \beta$ is valid. Thus, every interpretation $\mathcal{I}$ that satisfies α also satisfies β. □

Corollary 3.1: For all Σ-formulas α and β, we have $\vDash \alpha \to \beta$ iff $\alpha \to \beta$ is valid.

§3 OLD SLIDES

Syntax

□ Atomic Formulas

Atomic formulas are created by applying predicates to terms, such as $P(x)$ ("x satisfies property P") or $Q(f(x), y)$ ("$f(x)$ and y are related by Q").

Well-Formed Formulas (WFFs)

Formulas in FOL are defined inductively.

Base Case: Atomic formulas, such as $P(x)$, are WFFs.

Inductive Step: If α and β are WFFs, then the following are also WFFs:

- $\neg\alpha$
- $(\alpha \wedge \beta)$
- $(\alpha \vee \beta)$
- $(\alpha \rightarrow \beta)$
- $(\alpha \leftrightarrow \beta)$
- $(\forall x.\, \alpha)$
- $(\exists x.\, \alpha)$

Examples: $\forall x.\, (P(x) \rightarrow Q(x))$, $\exists y.\, (R(y) \wedge P(f(a, y)))$.

Semantics of First-Order Logic

The semantics of FOL specify how formulas are interpreted.

A **domain** (D) represents the set of all objects under consideration. **Variable assignments** map variables to elements in D, while **interpretations** assign meanings to constants, functions, and predicates.

Truth Conditions:

1. $P(t_1, ..., t_n)$ is true if $(t_1, ..., t_n)$ is in the interpretation of P.
2. $\forall x. \varphi$ is true if φ is true for all assignments $x \in D$.
3. $\exists x. \varphi$ is true if φ is true for some assignments $x \in D$.

Example: Let $D = \{1, 2, 3\}$, and $P(x)$ mean "x is even."

- $\forall x. P(x)$ is false (counter-example: $x = 3$ is not even).
- $\exists x. P(x)$ is true (example: $x = 2$ is even).

Examples

Suppose that P is a unary predicate, and Q is a binary predicate. Which of the following are true?

1. $\forall x.\, P(x) \vDash P(y)$
2. $P(x) \vDash \forall x.\, P(x)$
3. $\forall x.\, P(x) \vDash \exists y.\, P(y)$
4. $\exists x\, \forall y.\, Q(x, y) \vDash \forall y \exists x.\, Q(x, y)$
5. $\forall x\, \exists y.\, Q(x, y) \vDash \exists y \forall x.\, Q(x, y)$
6. $\vDash \exists x.\, (P(x) \to \forall y.\, P(y))$

Which models satisfy the following sentences?

Sentence	**Models**
$\forall x\, \forall y.\, (x = y)$	Models with a single element.
$\forall x\, \forall y.\, Q(x, y)$	Models (A, Q) where $Q = A \times A$.
$\forall x\, \exists y.\, Q(x, y)$	Models (A, Q) where $\operatorname{dom} Q = A$.

§4 Metatheory

Logical Theories

A logical theory T consists of a set of axioms (assumed true formulas) and all formulas derivable from them.

Examples:

- **Peano Arithmetic**: Defines natural numbers with axioms for addition and multiplication.
- **Group Theory**: Specifies algebraic structures with axioms for identity, inverses, and associativity.
- **Geometry**: Encodes relationships among points, lines, and planes.

A theory is **consistent** if no contradictions are derivable, and **complete** if every formula or its negation is derivable.

Logical Entailment

Logical entailment ($\Gamma \models \varphi$) means that φ is true in all models of Γ. For instance, if $\Gamma = \{\forall x.\, (P(x) \rightarrow Q(x)), P(a)\}$, then $\Gamma \models Q(a)$.

Entailment ensures that semantic truths are provable within a syntactic framework.

Proof Systems for First-Order Logic

Several formal proof systems are used to reason about FOL:

Natural Deduction: Introduces and eliminates quantifiers and connectives. For example, from $\forall x.\, P(x)$, infer $P(a)$ (universal elimination).

Sequent Calculus: Represents arguments as sequents ($\Gamma \to \Delta$), where Γ entails Δ. Example: From $\Gamma \to \forall x.\, P(x)$, infer $\Gamma \to P(a)$.

Resolution: Refutation-based method for proving unsatisfiability. It converts formulas into clausal form and derives contradictions to establish proofs.

Soundness and Completeness

Soundness ensures that every provable formula is true in all models ($\Gamma \vdash \varphi \rightarrow \Gamma \vDash \varphi$). Completeness guarantees that every formula true in all models is provable ($\Gamma \vDash \varphi \rightarrow \Gamma \vdash \varphi$).

The **Compactness Theorem** states that if every finite subset of Γ is satisfiable, then Γ is satisfiable. This is crucial for reasoning about infinite systems.

Decidability and Complexity

FOL is undecidable; no algorithm can determine the truth of every FOL formula. However, certain fragments, like monadic logic, are decidable. Semi-decidability allows verification of provable statements but not their refutations.

Theorem Proving and Applications

Automated theorem proving tools like Prover9, Coq, and Lean are widely used in:

- **Verification**: Ensuring correctness of algorithms.
- **Planning**: Reasoning about sequences of actions.
- **AI**: Modeling and querying knowledge bases.

§5 Advanced Topics

Model Theory

TODO:

- Structure
- Models

Proof Theory

TODO:

- Natural Deduction
- Sequent Calculus
- Resolution

§6 Applications

Modern Applications

TODO:

- Formal Verification
- Knowledge Representation
- Automated Reasoning
- Database Theory
- AI and Machine Learning

§7 Conclusion

Summary

First-order logic provides a foundation for reasoning about objects and their relationships. It extends propositional logic with quantifiers, predicates, and functions, enabling diverse applications in mathematics, AI, and formal verification. Understanding FOL is essential for exploring advanced topics like higher-order logic and model checking.

Exercises

- Parse tree of a FOL formula
- Describe Euclidean geometry as a FOL theory