

Formal Methods in Software Engineering

First-Order Logic, Spring 2026

Konstantin Chukharev

Introduction to FOL

Motivation: The Limits of Propositional Logic

Propositional logic was humanity's first formal calculus of reasoning (Boole, 1847; Frege, 1879). But it has fundamental limitations:

What PL cannot express:

- “All humans are mortal” (quantification)
- “Every even number > 2 is a sum of two primes” (Goldbach)
- “For all $\varepsilon > 0$, there exists $\delta > 0\dots$ ” (ε - δ proofs)
- “The array has no out-of-bounds access” (program verification)

Why this matters:

- Mathematics *needs* universal statements
- Program specifications quantify over inputs
- Database queries select objects satisfying properties
- Scientific theories make generalizations

What is First-Order Logic? The Historical Context

First-order logic emerged from three converging streams in late 19th/early 20th century:

Frege (1879): *Begriffsschrift* –
the first formal system with
quantifiers. Goal: foundations for
arithmetic. Notation was 2D and
hard to use.

Russell (1903): *Principia Mathematica* (with Whitehead,
1910-1913). Derived mathematics
from logic. Discovered Russell's
paradox: naive set theory is
inconsistent.

Hilbert (1920s): Hilbert's
program: prove consistency of
mathematics via finitary
methods. Pushed for formal
axiomatization. Goal shattered
by Gödel (1931).

Why “first-order”? The quantifiers $\forall x, \exists x$ range over *individuals* (elements of the domain).

In *second-order* logic, we could quantify over *sets* or *predicates*: $\forall P. \exists x. P(x)$.

In *higher-order* logic (used in Coq, Lean, Isabelle), we quantify over functions of functions, *etc.*

First-order logic is the *sweet spot*: expressive enough for most mathematics and verification, yet with nice metatheoretic properties (completeness, compactness).

Syntax: The Language of FOL

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*. Each language is specified by its *signature*.

Symbols of a first-order language are divided into *logical symbols* (fixed) and *non-logical parameters* (signature-dependent):

Logical symbols (universal):

- Parentheses: (,)
- Logical connectives: \neg , \wedge , \vee , \rightarrow , \iff
- Variables: x, y, z, \dots (infinite supply)
- Quantifiers: \forall (“for all”) and \exists (“there exists”)
- Equality: $=$ (sometimes, depending on convention)

Non-logical parameters (signature Σ):

- Constants: e.g., $0, \emptyset, e$ (arity 0 functions)
- Functions: e.g., S (successor), $+, \times, f(x, y)$
- Predicates: e.g., $<, \in, \text{Prime}(x), P(x, y)$

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

Note: Minimalism: Just \wedge, \neg suffice for connectives (others defined). Just \forall suffices for quantifiers ($\exists x.\varphi \equiv \neg\forall x.\neg\varphi$).

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 Languages

A first-order language is specified by its *parameters*.

Propositional logic:

- Equality: *no*
- Constants: *none*
- Predicates: A_1, A_2, \dots
- Functions: *none*

Set theory:

- Equality: *yes*
- Constants: \emptyset
- Predicates: \in
- Functions: *none*

Number theory:

- Equality: *yes*
- Constants: 0
- Predicates: $<$
- Functions: S (successor), $+$, \times

FOL Syntax

Signatures

Definition 1 (Signature): A *first-order signature* $\Sigma = \langle \mathcal{F}, \mathcal{R} \rangle$ consists of:

- A set of *function symbols* \mathcal{F} , each with an arity $n \geq 0$.
- A set of *relation symbols* (predicates) \mathcal{R} , each with an arity $n \geq 1$.

Functions of arity 0 are *constants*. We assume a countably infinite set of *variables* $\mathcal{V} = \{x, y, z, x_1, x_2, \dots\}$.

Example: Arithmetic: $\Sigma = \langle \{0, S, +, \times\}, \{<, =\} \rangle$

where 0 is a constant, S is unary, $+$, \times are binary functions; $<$, $=$ are binary relations.

Example: Graph theory: $\Sigma = \langle \emptyset, \{\text{Edge}, \text{Path}\} \rangle$

where Edge and Path are binary predicates (no function symbols).

Terms and Formulas

Definition 2 (Term): *Terms* over signature Σ are defined inductively:

- Every variable $x \in \mathcal{V}$ is a term.
- If $f \in \mathcal{F}$ has arity n and t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is a term.

Definition 3 (Formula): *Formulas* over Σ are defined inductively:

- If $R \in \mathcal{R}$ has arity n and t_1, \dots, t_n are terms, then $R(t_1, \dots, t_n)$ is an *atomic formula* (atom).
- If φ, ψ are formulas: $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \iff \psi)$ are formulas.
- If φ is a formula and $x \in \mathcal{V}$: $\forall x. \varphi$ and $\exists x. \varphi$ are formulas.

Example: $\forall x. (x = 0 \vee \exists y. S(y) = x)$ – “every natural number is 0 or a successor.”

Free and Bound Variables

The *scope* of $\forall x$ (or $\exists x$) is the immediately following subformula.

Definition 4 (Free Variables): The set $\text{FV}(\varphi)$ of *free variables* of φ is defined inductively:

- $\text{FV}(R(t_1, \dots, t_n)) = \text{Vars}(t_1) \cup \dots \cup \text{Vars}(t_n)$
- $\text{FV}(\neg\varphi) = \text{FV}(\varphi); \quad \text{FV}(\varphi \circ \psi) = \text{FV}(\varphi) \cup \text{FV}(\psi)$ for $\circ \in \{\wedge, \vee, \rightarrow, \iff\}$
- $\text{FV}(\forall x. \varphi) = \text{FV}(\exists x. \varphi) = \text{FV}(\varphi) \setminus \{x\}$

A formula with no free variables is a *sentence* (closed formula).

Example: In $\forall x. P(x, y)$: x is *bound* (in scope of $\forall x$), y is *free*. Not a sentence.

Note: Only *sentences* have a definite truth value in a structure. Formulas with free variables are like “open predicates” — they become true/false once you fix values for the free variables.

Substitution

Definition 5 (Substitution): $\varphi[t/x]$ denotes the formula obtained by replacing every *free* occurrence of x with term t .

A substitution $[t/x]$ is *capture-free* in φ if no free variable in t becomes bound.

Example: $\forall y. (x < y) [S(0)/x] = \forall y. (S(0) < y)$ – correct substitution.

$\exists y. (x < y) [y + 1/x] = \exists y. (y + 1 < y)$ – **variable capture!**

The free y in the substituted term becomes bound. Fix: rename bound variable first: $\exists z. (y + 1 < z)$.

Variable capture: Always check for name clashes before substituting. Rename bound variables if necessary.

FOL Semantics

Structures (Models)

Definition 6 (Structure (Model)): A *structure* \mathfrak{A} for signature $\Sigma = \langle \mathcal{F}, \mathcal{R} \rangle$ consists of:

- A non-empty *domain* A – the set of objects we reason about.
- For each n -ary function symbol $f \in \mathcal{F}$: a function $f^{\mathfrak{A}} : A^n \rightarrow A$.
- For each n -ary relation symbol $R \in \mathcal{R}$: a relation $R^{\mathfrak{A}} \subseteq A^n$.

A *variable assignment* $\sigma : \mathcal{V} \rightarrow A$ maps each variable to a domain element.

Example: For the arithmetic signature $\Sigma = \langle \{0, S, +, \times\}, \{<, =\} \rangle$:

- $\mathfrak{N} = (\mathbb{N}, 0, S, +, \times, <, =)$ – the *standard* (intended) model.
- $\mathfrak{Z} = (\mathbb{Z}, 0, S, +, \times, <, =)$ – a different, equally valid structure.
- The sentence $\forall x. \exists y. x = S(y) \vee x = 0$ is true in \mathfrak{N} (every natural number is 0 or a successor) but *false* in \mathfrak{Z} (take $x = -1$: it is not 0 and not a successor of any integer under the standard successor).

This illustrates a key point: the same sentence can be true in one structure and false in another. *Validity* means truth in *all* structures.

Evaluating FOL Formulas

Given a structure \mathfrak{A} and a variable assignment σ , truth is defined inductively. The key ingredient compared to PL: quantifiers range over domain elements.

Definition 7 (Satisfaction Relation): The relation $\mathfrak{A}, \sigma \models \varphi$ is defined inductively:

- $\mathfrak{A}, \sigma \models R(t_1, \dots, t_n)$ iff $(t_1^{\mathfrak{A}, \sigma}, \dots, t_n^{\mathfrak{A}, \sigma}) \in R^{\mathfrak{A}}$
- Boolean connectives: as in PL.
- $\mathfrak{A}, \sigma \models \forall x. \varphi$ iff $\mathfrak{A}, \sigma[x \mapsto a] \models \varphi$ for *every* $a \in A$.
- $\mathfrak{A}, \sigma \models \exists x. \varphi$ iff $\mathfrak{A}, \sigma[x \mapsto a] \models \varphi$ for *some* $a \in A$.

Here $\sigma[x \mapsto a]$ maps x to a and agrees with σ on all other variables.

For *sentences* (no free variables), the truth value depends only on the structure: we write $\mathfrak{A} \models \varphi$.

Example: Let $\mathfrak{A} = (\{1, 2, 3\}, <^{\mathfrak{A}} = \{(1, 2), (1, 3), (2, 3)\})$ be a structure for signature $\{<\}$.

Evaluate $\forall x. \exists y. x < y$ in \mathfrak{A} :

- $x = 1$: need y with $1 < y$. Take $y = 2$. ✓

Evaluating FOL Formulas [2]

- $x = 2$: need y with $2 < y$. Take $y = 3$. ✓
- $x = 3$: need y with $3 < y$. No such y in $\{1, 2, 3\}$. ✗

Result: $\mathfrak{A} \not\models \forall x. \exists y. x < y$. The formula is *falsified* by the element 3.

But $\forall x. \exists y. x < y$ *is* satisfied in $\mathfrak{N} = (\mathbb{N}, <)$ – for every n , take $y = n + 1$.

Validity and Satisfiability in FOL

Definition 8: Let φ be an FOL sentence.

- φ is *valid* ($\models \varphi$) if $\mathfrak{A} \models \varphi$ for *every* structure \mathfrak{A} .
- φ is *satisfiable* if $\mathfrak{A} \models \varphi$ for *some* structure \mathfrak{A} .
- φ is *unsatisfiable* if no structure satisfies it.

Critical difference from PL:

- In PL, the space of interpretations is *finite* (2^n truth assignments).
- In FOL, structures can have *infinite* domains of *any* cardinality — decision procedures are fundamentally harder, sometimes even impossible (undecidable).

Example:

- $\forall x. (P(x) \vee \neg P(x))$ — valid (instance of LEM, holds in every structure).
- $\forall x. P(x) \wedge \exists x. \neg P(x)$ — unsatisfiable.
- $\exists x. P(x)$ — satisfiable (in any non-empty structure with P non-empty) but not valid.

Quantifier Equivalences

Many useful equivalences govern quantifiers:

De Morgan duality for quantifiers:

- $\neg \forall x. \varphi \equiv \exists x. \neg\varphi$
- $\neg \exists x. \varphi \equiv \forall x. \neg\varphi$

Distribution:

- $\forall x. (\varphi \wedge \psi) \equiv (\forall x. \varphi) \wedge (\forall x. \psi)$
- $\exists x. (\varphi \vee \psi) \equiv (\exists x. \varphi) \vee (\exists x. \psi)$

Vacuous quantification (if x not free in ψ):

- $\forall x. \psi \equiv \psi, \quad \exists x. \psi \equiv \psi$

Commutativity of like quantifiers:

- $\forall x. \forall y. \varphi \equiv \forall y. \forall x. \varphi$
- $\exists x. \exists y. \varphi \equiv \exists y. \exists x. \varphi$

Prenex pulling (if x not free in ψ):

- $\psi \wedge \forall x. \varphi \equiv \forall x. (\psi \wedge \varphi)$
- $\psi \vee \exists x. \varphi \equiv \exists x. (\psi \vee \varphi)$

Quantifier Equivalences [2]

$\exists x. \forall y$ is generally *stronger* than $\forall y. \exists x$ – the former asserts a *single* x for *all* y .

Example: Let $R(x, y) \equiv "x \text{ is the parent of } y"$. Then:

- $\exists x. \forall y. R(x, y)$ = “someone is the parent of *everyone*” – false.
- $\forall y. \exists x. R(x, y)$ = “everyone *has* a parent” – true.

In general: $\exists A \models \forall E$, but $\forall E \not\models \exists A$.

Prenex Normal Form

Definition 9 (Prenex Normal Form (PNF)): A formula is in *prenex normal form* if it has the shape:

$$Q_1 x_1 \cdot Q_2 x_2 \cdot \dots \cdot Q_n x_n \cdot \psi$$

where each $Q_i \in \{\forall, \exists\}$ and ψ is *quantifier-free* (the *matrix*).

Theorem 1: Every FOL formula can be converted to an equivalent formula in PNF (after renaming bound variables to avoid capture if necessary).

Example: $\forall x. P(x) \rightarrow \exists y. Q(x, y)$
 $\equiv \forall x. (\neg P(x) \vee \exists y. Q(x, y))$ (eliminate \rightarrow)
 $\equiv \forall x. \exists y. (\neg P(x) \vee Q(x, y))$ (pull $\exists y$ out)

PNF separates the *quantifier prefix* (alternation structure) from the *propositional skeleton* (matrix).
The alternation depth ($\forall \exists \forall \dots$) determines complexity.

FOL Proofs

FOL Proof Rules: Quantifiers

\forall -introduction ($\forall i$):

Derive $\varphi(y)$ for an *arbitrary* y .

Conclude $\forall x. \varphi(x)$.

1	$\varphi(y)$	$\vdots (y \text{ arbitrary})$
2	$\forall x. \varphi(x)$	$\forall i 1$

\exists -introduction ($\exists i$):

From $\varphi(t)$ for some term t , conclude $\exists x. \varphi(x)$.

1	$\varphi(t)$	<i>Premise</i>
2	$\exists x. \varphi(x)$	$\exists i 1$

\forall -elimination ($\forall e$):

From $\forall x. \varphi(x)$, conclude $\varphi(t)$ for *any* term t .

1	$\forall x. \varphi(x)$	<i>Premise</i>
2	$\varphi(t)$	$\forall e 1$

\exists -elimination ($\exists e$):

From $\exists x. \varphi(x)$, open a subproof assuming $\varphi(y)$ for *fresh* y , derive ψ . Conclude ψ .

1	$\exists x. \varphi(x)$	<i>Premise</i>
2	$\varphi(y)$	$y \text{ fresh}$
3	\vdots	\vdots
4	ψ	\vdots
5	ψ	$\exists e 1, 2-4$

FOL Proof Rules: Quantifiers [2]

Side conditions:

- $\forall i$: y must be *arbitrary* – not free in any undischarged assumption.
- $\forall e$: t can be any term (universal *instantiation*).
- $\exists i$: give a *witness* term t .
- $\exists e$: y must be *fresh* – not free in ψ or any undischarged assumption besides $\varphi(y)$.

FOL Proof: A Complete Fitch Example

Example: Prove: $\forall x. (P(x) \rightarrow Q(x)), \exists x. P(x) \vdash \exists x. Q(x)$

"If every P is a Q , and some P exists, then some Q exists."

1	$\forall x. (P(x) \rightarrow Q(x))$	Premise	Line 3: open $\exists e$ subproof – name the witness a (fresh).
2	$\exists x. P(x)$	Premise	Line 4: instantiate $\forall x. (P(x) \rightarrow Q(x))$ with a ($\forall e$).
3	$P(a)$	a fresh	Line 5: modus ponens on lines 3 and 4.
4	$P(a) \rightarrow Q(a)$	$\forall e 1$	Line 6: from $Q(a)$, existentially generalize ($\exists i$).
5	$Q(a)$	$\rightarrow e 3, 4$	Line 7: close $\exists e$ – the conclusion $\exists x. Q(x)$ does not mention a , so it survives.
6	$\exists x. Q(x)$	$\exists i 5$	
7	$\exists x. Q(x)$	$\exists e 2, 3-6$	

Semantic Arguments for FOL

Semantic tableaux rules extend to FOL with rules for quantifiers:

$$(a) \frac{\mathcal{I} \models \neg\alpha}{\mathcal{I} \not\models \alpha}$$

$$(b) \frac{\mathcal{I} \not\models \neg\alpha}{\mathcal{I} \models \alpha}$$

$$(c) \frac{\mathcal{I} \models \alpha \wedge \beta}{\mathcal{I} \models \alpha, \mathcal{I} \models \beta}$$

$$(d) \frac{\mathcal{I} \not\models \alpha \wedge \beta}{\mathcal{I} \not\models \alpha \mid \mathcal{I} \not\models \beta}$$

$$(e) \frac{\mathcal{I} \models \alpha \vee \beta}{\mathcal{I} \models \alpha \mid \mathcal{I} \models \beta}$$

$$(f) \frac{\mathcal{I} \not\models \alpha \vee \beta}{\mathcal{I} \not\models \alpha, \mathcal{I} \not\models \beta}$$

$$(i) \frac{\mathcal{I} \models \alpha \quad \mathcal{I} \not\models \alpha}{\mathcal{I} \models \perp}$$

$$(g) \frac{\mathcal{I} \models \alpha \rightarrow \beta}{\mathcal{I} \not\models \alpha \mid \mathcal{I} \models \beta}$$

$$(h) \frac{\mathcal{I} \not\models \alpha \rightarrow \beta}{\mathcal{I} \models \alpha, \mathcal{I} \not\models \beta}$$

$$(j) \frac{\mathcal{I} \models \alpha \iff \beta}{\mathcal{I} \models \alpha, \mathcal{I} \models \beta \mid \mathcal{I} \not\models \alpha, \mathcal{I} \not\models \beta}$$

$$(k) \frac{\mathcal{I} \not\models \alpha \iff \beta}{\mathcal{I} \not\models \alpha, \mathcal{I} \models \beta \mid \mathcal{I} \models \alpha, \mathcal{I} \not\models \beta}$$

Metatheorems and the Limits of Logic

FOL Soundness and Completeness

Theorem 2 (Gödel's Completeness Theorem (1930)): FOL with standard proof rules is both *sound* and *complete*:

$$\Gamma \vdash \varphi \iff \Gamma \vDash \varphi$$

Proof: (*Soundness.*) Each rule preserves truth: if premises hold in \mathfrak{A} , so does the conclusion.

(*Completeness, sketch.*) If $\Gamma \not\vdash \varphi$, then $\Gamma \cup \{\neg\varphi\}$ is consistent. Extend to a maximally consistent set Γ^* (Lindenbaum's lemma). Add Henkin witnesses for existential formulas. Build a canonical model from equivalence classes of closed terms. This model satisfies $\Gamma \cup \{\neg\varphi\}$, so $\Gamma \not\models \varphi$. Contrapositive gives the result. □

Not the “incompleteness theorem” – here “complete” means the proof system derives everything semantically true in *all* structures.

FOL Validity is Undecidable

Despite completeness, there is *no algorithm* that always terminates and correctly decides FOL validity.

Theorem 3 (Church–Turing Theorem (1936)): The validity problem for FOL is *undecidable*: no Turing machine can decide, given an arbitrary FOL sentence φ , whether $\models \varphi$.

Decidable (PL):

- Finite search space (2^n interpretations)
- Truth tables always terminate
- SAT is NP-complete, VALID is co-NP-complete

Undecidable (FOL):

- Infinite/unbounded structures
- Proof search may not terminate
- Valid = semi-decidable (r.e.)
- Satisfiable = semi-decidable (co-r.e.)

FOL validity is **semi-decidable**: if φ is valid, proof search *will* find a proof (by completeness). If φ is not valid, search may run forever.

SMT solvers restrict to *decidable fragments* of FOL – theories where satisfiability *can* be decided.

The Compactness Theorem

Theorem 4 (Compactness Theorem): A (possibly infinite) set of FOL sentences Γ is satisfiable if and only if every *finite* subset of Γ is satisfiable.

Proof (sketch): (\Rightarrow) Trivial: any model of Γ satisfies every finite subset.

(\Leftarrow) If Γ is unsatisfiable, then by completeness there is a proof of \perp from Γ . Every proof uses only *finitely many* premises, so some finite $\Gamma_0 \subseteq \Gamma$ is already unsatisfiable. \square

Example: **Non-standard models of arithmetic:** Let $\Gamma = \text{Th}(\mathbb{N}) \cup \{c > 0, c > 1, c > 2, \dots\}$ where c is a fresh constant. Every finite subset is satisfiable (interpret c as a large enough number). By compactness, Γ is satisfiable – in a model with an “infinite” element c larger than all standard naturals. This is a *non-standard model* of arithmetic.

If a specification has a bug (is unsatisfiable), some *finite* subset of constraints already witnesses it – this is why *bounded model checking* works: check finitely many constraints at a time.

The Löwenheim–Skolem Theorem

Theorem 5 (Löwenheim–Skolem Theorem): If an FOL sentence (or countable set of sentences) has an *infinite* model, then it has a model of *every* infinite cardinality.

Skolem's paradox (1922): ZFC proves uncountable sets exist, yet by Löwenheim–Skolem, ZFC has a *countable* model. Resolution: “uncountable” is *relative* to the model’s membership relation.

Expressive limitations of FOL (compactness + Löwenheim–Skolem):

- Cannot define “exactly the natural numbers” (up to isomorphism).
- Cannot express “the domain is finite” or “the domain is countable.”
- Cannot distinguish between structures of different infinite cardinalities.

These limitations motivate *stronger* logics (second-order, infinitary) and *decidable* fragments (monadic FOL, EPR, SMT theories).

You cannot write an FOL spec that pins down “exactly the integers.”

SMT solvers work around this by *fixing* the interpretation of theory symbols ($+$, \times , $<$) – they reason about a *specific* structure, not all possible models.

Gödel's Incompleteness Theorems

Completeness (above): “if φ is true in *all* structures, it is provable.” *Incompleteness* (below): “if we fix *one* structure (\mathbb{N}), some true sentences are unprovable.” These concern different questions.

Theorem 6 (First Incompleteness Theorem): Any *consistent* formal system \mathcal{T} capable of expressing elementary arithmetic contains sentences that are *true* (in the standard model \mathbb{N}) but *unprovable* in \mathcal{T} .

Theorem 7 (Second Incompleteness Theorem): If \mathcal{T} is consistent and sufficiently powerful, then \mathcal{T} *cannot prove its own consistency*:

$$\mathcal{T} \not\vdash \text{Con}(\mathcal{T})$$

Sufficiently powerful: \mathcal{T} must be capable of representing all computable functions – essentially, \mathcal{T} must contain Robinson arithmetic (Q) or stronger.

Gödel's *completeness* theorem: FOL proof systems are complete w.r.t. semantic consequence. His *incompleteness* theorems: specific *theories* (like arithmetic) have true-but-unprovable sentences. Different notions of “completeness”!

Incompleteness: The Key Idea

The proof relies on *self-reference*, made mathematically precise via Gödel numbering.

Every formula, proof, and syntactic operation is encoded as a natural number. There is an arithmetic formula $\text{Prov}(n)$ saying “ n is the Gödel number of a provable sentence.” Construct a sentence G :

$$G \equiv \text{"I am not provable in } \mathcal{T}\text{"}$$

If G is provable in \mathcal{T} :

- \mathcal{T} proves G
- G says “ G is not provable”
- So \mathcal{T} proves something false
- Contradicts *consistency* of \mathcal{T}

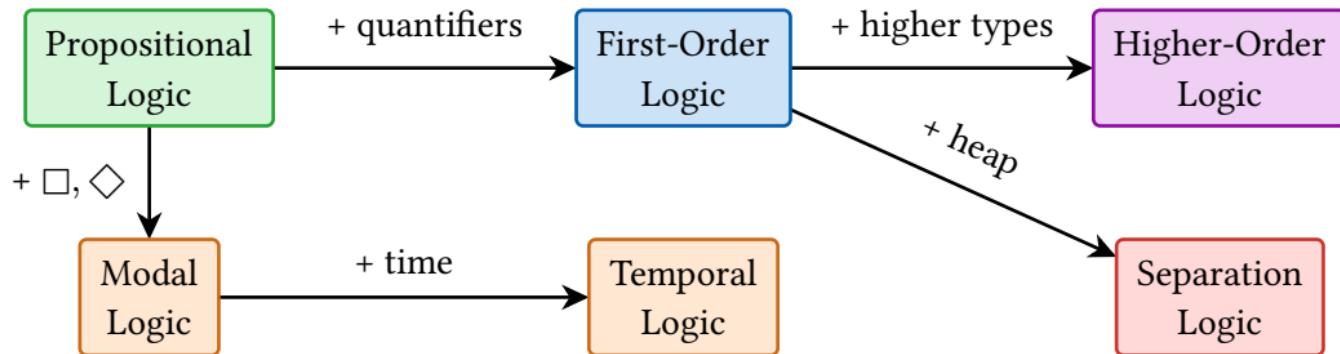
If G is not provable:

- G 's assertion is *true*
- So G is true but unprovable
- \mathcal{T} is *incomplete*

Incompleteness means *no* verification system can prove *all* true program properties. In practice, automated tools are remarkably effective for *specific* programs.

The Landscape of Logics

Classical PL and FOL are two points in a rich space of formal systems. Different verification tasks need different logics:



Modal Logic: $\Box\varphi$ (“necessarily”) and $\Diamond\varphi$ (“possibly”). Kripke semantics: worlds + accessibility.

Temporal Logic: LTL ($\Box, \Diamond, \mathcal{U}$), CTL, CTL*. Model checking: $\Box\Diamond$ response.

Separation Logic: Spatial connectives $(*, *)$ for heap memory. Powers Meta Infer, VeriFast.

Decidability Landscape

Computational complexity of logical decision problems:

Logic / Fragment	SAT	Validity	Complexity
Propositional	Decidable	Decidable	NP-c / co-NP-c
Modal (K, S4, S5)	Decidable	Decidable	PSPACE-complete
FOL (general)	Undecidable	Undecidable	Semi-decidable
FOL monadic	Decidable	Decidable	NEXPTIME-complete
Presburger ($\mathbb{N}, +$)	Decidable	Decidable	2-EXPTIME
Arithmetic ($\mathbb{N}, +, \times$)	Undecidable	Undecidable	Not even semi-dec.

Why Theories?

FOL validity is *undecidable* in general. But verification does not need *all* of FOL — it needs to reason about *specific* domains: integers, arrays, bit-vectors, *etc.*

Definition 10 (First-Order Theory): A *first-order theory* \mathcal{T} is a set of FOL sentences (axioms) over a fixed signature Σ . A \mathcal{T} -*model* is a structure satisfying all axioms in \mathcal{T} . \mathcal{T} -satisfiability asks: is there a \mathcal{T} -*model* satisfying a given formula?

Example:

- *Theory of linear integer arithmetic* (LIA): $\Sigma = \{0, 1, +, <, =\}$. Only structures isomorphic to \mathbb{Z} considered.
- *Theory of arrays*: $\Sigma = \{\text{read}, \text{write}\}$ with McCarthy axioms.
- *Theory of equality with uninterpreted functions* (EUF): $\Sigma = \{=, f_1, f_2, \dots\}$, congruence axioms only.

By *fixing* the theory, we restrict to structures where decision procedures *can* terminate.

From English to FOL

Translating natural language to FOL formulas is a key skill. Here are worked examples over the signature of arithmetic: $\Sigma = \langle \{0, S, +, \times\}, \{<, =\} \rangle$.

1. *There is no largest natural number.*

$$\neg \exists x. \forall y. y \leq x \quad (\text{equivalently: } \forall x. \exists y. x < y)$$

2. *Every prime greater than 2 is odd.*

$$\forall p. (\text{Prime}(p) \wedge p > 2) \rightarrow \text{Odd}(p)$$

3. *For every natural number there is a greater one.*

$$\forall x. \exists y. x < y$$

4. *Two distinct natural numbers cannot have the same successor.*

$$\forall x. \forall y. \neg(x = y) \rightarrow \neg(S(x) = S(y))$$

5. *There are at least two natural numbers smaller than 3.*

$$\exists x. \exists y. \neg(x = y) \wedge x < S(S(S(0))) \wedge y < S(S(S(0)))$$

From English to FOL [2]

Tip: When formalizing, identify (1) the domain, (2) the quantifier structure, and (3) the predicate/function symbols needed. Common pitfall: confusing the direction of implication after \forall .

Exercises

Exercises: FOL Syntax and Semantics

1. Formalize the following in FOL:
 - “Every prime number greater than 2 is odd.”
 - “There is no largest natural number.”
 - “If f is injective and $A \subseteq B$, then $f(A) \subseteq f(B)$.”
2. Determine free and bound variables, and whether each formula is a sentence:
 - $\forall x. (P(x) \rightarrow \exists y. Q(x, y))$
 - $\exists x. (x = y + 1)$
 - $\forall x. \forall y. (R(x, y) \rightarrow R(y, x))$
3. Convert to prenex normal form: $(\forall x. P(x)) \rightarrow (\exists y. Q(y))$
4. Explain informally: why does $\exists x. \forall y. R(x, y) \vDash \forall y. \exists x. R(x, y)$ hold, but $\forall y. \exists x. R(x, y) \not\vDash \exists x. \forall y. R(x, y)$? Give a concrete counterexample for the latter.

Exercises: FOL Proofs and Metatheorems

1. Prove the following using natural deduction (Fitch notation):
 - $\forall x. (P(x) \rightarrow Q(x)), \exists x. P(x) \vdash \exists x. Q(x)$
 - $\forall x. (P(x) \rightarrow Q(x)) \vdash (\exists x. P(x)) \rightarrow (\exists x. Q(x))$
 - $\forall x. P(x), \forall x. (P(x) \rightarrow Q(x)) \vdash \forall x. Q(x)$
2. Construct a countermodel to show that $\exists x. P(x) \wedge \exists x. Q(x) \not\models \exists x. (P(x) \wedge Q(x))$.
3. ★ Using compactness, show that “the domain is finite” cannot be expressed by any *single* FOL sentence (or even by any *set* of FOL sentences). *Hint:* Consider the set of sentences φ_n asserting “there exist at least n distinct elements.”
4. ★ The compactness theorem fails for *second-order* logic. Explain why, and give an example of a property SOL can express but FOL cannot.