First-Order Logic: Languages, Structures, and Truth

Advanced Logic

Contents

1 Introduction: Why Languages and Structures?

Model theory studies the relation between *formal languages* and *mathematical structures*. In first-order logic, a language provides symbols and syntactic rules, while a structure provides *interpretations* for those symbols.

Philosophically, this framework allows us to:

- separate syntax from semantics,
- give precise truth-conditions for sentences,
- understand reference independently of psychology or epistemology.

A key idea throughout is that *truth is always relative to a structure*.

1.1 Signatures and Non-Logical Vocabulary

A *signature* $\mathcal{L}$ is a (possibly empty) set of non-logical symbols. It determines *what kind of things the language can talk about*.

1.1.1 Kinds of Symbols

- **Constant symbols**: intended to name specific objects.

  Example: 0 in arithmetic.

- **Relation symbols**: intended to express properties or relations.

  Example: $<$ as a binary relation.

- **Function symbols**: intended to express operations.

  Example: $+$, $\cdot$, or the successor symbol S.

The signature fixes the expressive resources of the language, but *not* their meaning.

1.2 $\mathcal{L}$-Structures

1.2.1 Definition

An $\mathcal{L}$-structure (or model) is a pair

$$\mathcal{M} = \langle D, I \rangle$$

where:

- D is a non-empty set (the *domain*);
- I is an interpretation function.

The interpretation function assigns meanings to symbols:

- constants $\mapsto$ elements of D,
- n-ary relations $\mapsto$ subsets of D^n,
- n-ary functions $\mapsto$ functions from D^n to D.

1.2.2 Illustration

Let $\mathcal{L} = \{0, S, +\}$. The *standard model of arithmetic* is:

$$\mathcal{M} = (\mathbb{N}, 0, S, +)$$

where:

- $I(0) = 0$,
- $I(S)(n) = n + 1$,
- $I(+)(n, m) = n + m$.

1.3 Relations and Functions

1.3.1 Relations

An n-ary relation on a set A is a subset of A^n.

For example, the less-than relation on $\mathbb{N}$ is:

$$\{\langle x, y \rangle \mid x < y\}.$$

Writing $x < y$ is shorthand for $\langle x, y \rangle \in I(<)$.

1.3.2 Functions

A function $f : A \to B$ assigns *exactly one* output to each input.

This functional character is crucial: unlike relations, functions cannot be ambiguous.

1.4 Object Language vs Meta-language

There is an important conceptual distinction between:

- symbols of the formal language,
- objects in the structure.

For example:

$$I(0) = 0$$

Here, the symbol 0 belongs to the object language, while the number 0 belongs to the meta-language.

In practice, we usually blur this distinction to avoid cumbersome notation.

2 First-Order Languages

A first-order language $\mathcal{L}$ contains:

- non-logical symbols from the signature,
- logical symbols:
  - variables $x, y, z, \ldots$,
  - quantifiers $\forall, \exists$,
  - identity $=$,
  - connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$,
  - parentheses.

2.1 Terms

Terms are expressions intended to *refer to objects*.

2.1.1 Definition

The set of terms is defined recursively:

- constants are terms,
- variables are terms,
- if f is n-ary and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

2.1.2 Illustrations

In arithmetic:

- 0 is a term,
- x is a term,
- $S(0)$ is a term,
- $+(S(0), x)$ is a term.

Crucially, *terms are not true or false*: they denote objects.

2.2 Atomic Formulae

Atomic formulae are the simplest truth-evaluable expressions.

- $t_i = t_j$,
- $R(t_1, \ldots, t_n)$.

Example:

$$S(0) = S(0), \quad < (0, S(0)).$$

2.3 Well-Formed Formulae

Complex formulae are built from atomic ones using:

- connectives,
- quantifiers.

Quantifiers bind variables:

$$\forall x \varphi, \quad \exists x \varphi.$$

3 Free and Bound Variables

3.1 In Terms

The set $\mathrm{var}(t)$ of free variables in a term:

- $\mathrm{var}(c) = \emptyset$,
- $\mathrm{var}(x) = \{x\}$,
- $\mathrm{var}(f(t_1, \ldots, t_n)) = \bigcup \mathrm{var}(t_i)$.

3.2 In Formulae

A variable occurrence is *bound* if it falls under a quantifier.

3.3 Illustrations

- $\exists y\, Rxy$: x is free, y is bound.
- $\forall x \exists y\, Rxy$: no free variables.

Closed formulae (sentences) express complete claims.

3.4 Free Variables in a Formula: An Inductive Definition

We now define one of the most important syntactic notions in first-order logic: the set of *free variables* of a formula. This notion plays a crucial role in the semantics of quantification and will also serve as our first clear example of an *inductive definition*.

3.4.1 What Is an Inductive Definition?

An *inductive definition* (also called a *recursive definition*) is a definition that proceeds by:

- specifying the value of a notion for the *simplest cases* (the base cases),
- explaining how to compute its value for *more complex cases*, assuming it has already been defined for simpler components.

Inductive definitions are especially well suited for syntactic objects such as:

- terms,
- formulae,
- proofs.

This is because such objects are themselves defined recursively: complex formulae are built step by step from simpler ones.

3.4.2 Why the Definition of Free Variables Is Inductive

Formulae of first-order logic are generated inductively:

- atomic formulae are the simplest cases;
- more complex formulae are obtained by applying connectives and quantifiers.

Therefore, any notion defined *on formulae*—such as the set of free variables—must mirror this structure. The definition of free variables follows the construction of formulae exactly, clause by clause.

3.4.3 Definition of the Set of Free Variables

We define, by induction on the structure of formulae, the set $\mathrm{Varf}(\varphi)$ of free variables occurring in a formula φ.

- **Atomic formulae:** If φ is an atomic formula $R(t_1, \ldots, t_n)$ or $t_i = t_j$, then:
$$\mathrm{Varf}(\varphi) = \mathrm{var}(t_1) \cup \cdots \cup \mathrm{var}(t_n).$$
That is, the free variables of an atomic formula are exactly the variables that occur free in its constituent terms.
- **Negation:** If $\varphi = \neg\psi$, then:
$$\mathrm{Varf}(\varphi) = \mathrm{Varf}(\psi).$$
- **Binary connectives:** If $\varphi = \psi \circ \theta$, where $\circ$ is one of $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, then:
$$\mathrm{Varf}(\varphi) = \mathrm{Varf}(\psi) \cup \mathrm{Varf}(\theta).$$
- **Quantifiers:** If $\varphi = \forall x\psi$ or $\varphi = \exists x\psi$, then:
$$\mathrm{Varf}(\varphi) = \mathrm{Varf}(\psi) \setminus \{x\}.$$
Intuitively, the quantifier binds all free occurrences of x in ψ.

No other cases are allowed: this exhausts all possible well-formed formulae.

3.5 Open and Closed Formulae

- A formula φ is called a *closed formula* (or a *sentence*) if:

$$\mathrm{Varf}(\varphi) = \emptyset.$$

- A formula is called *open* if it has at least one free variable.

Illustrations

- $\exists y\, Rxy$ is an open formula, since:

$$\mathrm{Varf}(\exists y\, Rxy) = \{x\}.$$

- $\forall x \exists y\, Rxy$ is a closed formula, since:

$$\mathrm{Varf}(\forall x \exists y\, Rxy) = \emptyset.$$

Pedagogical remark. This inductive definition guarantees that the set of free variables of a formula is *well-defined*: every formula has a unique syntactic decomposition, and the definition tells us exactly how to compute $\mathrm{Varf}(\varphi)$ by following that decomposition step by step.

4 Tarski's Truth Definition for Formulae

4.1 Why Assignments Are Needed

Consider the open formula:

$$x + 1 = 2.$$

Its truth depends on what x refers to. We therefore introduce assignments.

4.1.1 Variable Assignments

A variable assignment σ assigns an element of D to each variable.

Example:

$$\sigma(x) = 1, \quad \sigma(y) = 3.$$

4.1.2 Interpreting Terms Relative to an Assignment

We define $I_\sigma(t)$ recursively:

- $I_\sigma(x) = \sigma(x)$,
- $I_\sigma(c) = I(c)$,
- $I_\sigma(f(t_1, \ldots, t_n)) = I(f)(I_\sigma(t_1), \ldots, I_\sigma(t_n))$.

4.1.3 Illustration

Let $\sigma(x) = 1$. Then:

$$I_\sigma(S(x)) = 2.$$

4.2 Truth for Atomic Formulae

- $\mathcal{M} \vDash_\sigma t_i = t_j$ iff $I_\sigma(t_i) = I_\sigma(t_j)$;
- $\mathcal{M} \vDash_\sigma R(t_1, \ldots, t_n)$ iff $\langle I_\sigma(t_1), \ldots, I_\sigma(t_n)\rangle \in I(R)$.

4.3 Tarski's Truth Definition for Propositional Formulae

Before defining the semantics of quantifiers, we must specify the truth conditions for formulae built using propositional connectives. These clauses extend the semantic evaluation from atomic formulae to all quantifier-free formulae.

Truth is defined relative to a structure $\mathcal{M}$ and a variable assignment σ.

4.3.1 Contradiction

The contradiction symbol $\bot$ is never true:

$$\mathcal{M} \vDash_\sigma \bot \quad \text{never holds.}$$

Intuitively, $\bot$ represents a formula that is false in every structure and under every assignment.

4.3.2 Negation

If φ is a formula, then:

$$\mathcal{M} \vDash_\sigma \neg\varphi \quad \text{iff} \quad \mathcal{M} \nvDash_\sigma \varphi.$$

Negation reverses the truth value of the formula it applies to.

4.3.3 Conjunction

If φ and ψ are formulae, then:

$$\mathcal{M} \vDash_\sigma (\varphi \wedge \psi) \quad \text{iff} \quad \mathcal{M} \vDash_\sigma \varphi \text{ and } \mathcal{M} \vDash_\sigma \psi.$$

A conjunction is true exactly when both conjuncts are true.

4.3.4 Disjunction

If φ and ψ are formulae, then:

$$\mathcal{M} \vDash_\sigma (\varphi \vee \psi) \quad \text{iff} \quad \mathcal{M} \vDash_\sigma \varphi \text{ or } \mathcal{M} \vDash_\sigma \psi.$$

A disjunction is true when at least one of its disjuncts is true.

4.3.5 Conditional

If φ and ψ are formulae, then:

$$\mathcal{M} \vDash_\sigma (\varphi \to \psi) \quad \text{iff} \quad \mathcal{M} \nvDash_\sigma \varphi \text{ or } \mathcal{M} \vDash_\sigma \psi.$$

Thus, a conditional is true if its antecedent is false or if its consequent is true.

4.4 Quantifiers

4.4.1 Substitutional Semantics for Quantifiers

A natural idea—often introduced first in philosophy—is *substitutional semantics*:

> $\forall x\, Fx$ is true if and only if Fa is true for every constant symbol a.

This idea treats quantification as ranging over *names* rather than objects.

4.4.2 Why This Is Attractive

- It avoids commitment to abstract domains.
- It aligns with an intuitive linguistic picture: quantify over expressions.
- It does not require variable assignments.

4.4.3 Why It Fails in General

Despite its appeal, substitutional semantics is inadequate for first-order logic:

- A language may have *fewer constant symbols than objects* in the domain.
- Many structures (e.g. $\mathbb{R}$) are uncountable.
- First-order logic does not require a name for every object.

Thus, even if Fa is true for every constant a, there may still be unnamed objects that do not satisfy F.

Philosophical moral: substitution over symbols cannot replace quantification over objects.

4.5 Tarski's Objectual Semantics

Tarski's solution is to interpret quantifiers as ranging over *objects in the domain*, not symbols in the language.

This requires introducing *variable assignments*.

4.5.1 Variants of Assignments

Given a variable x and an object o, $\sigma[x \to o]$ is the assignment that:

- sends x to o,
- agrees with σ on all other variables.

This allows quantifiers to temporarily "test" different objects.

4.5.2 Truth for Quantified Formulae

Universal Quantifier

$$\mathcal{M} \vDash_\sigma \forall x \varphi$$

iff for all $o \in D$,

$$\mathcal{M} \vDash_{\sigma[x \to o]} \varphi.$$

Existential Quantifier

$$\mathcal{M} \vDash_\sigma \exists x \varphi$$

iff for some $o \in D$,

$$\mathcal{M} \vDash_{\sigma[x \to o]} \varphi.$$

Illustration $\forall x\, x = x$ is true because *every object is identical to itself.* $\exists x\, x = 0$ is true because *some object* (namely 0) satisfies the condition.

4.6 Truth in a Model

A sentence φ is true in $\mathcal{M}$ iff:

$$\mathcal{M} \vDash \varphi \quad \text{iff} \quad \text{for all assignments } \sigma,\ \mathcal{M} \vDash_\sigma \varphi.$$

Since sentences have no free variables, assignments no longer matter.

This completes Tarski's semantic definition of truth for first-order logic.