

Automated Theorem Proving

Lecture 4: First-Order Logic

Prof. Dr. Jasmin Blanchette

based on slides by Dr. Uwe Waldmann

Winter Term 2025/26

Part 3: First-Order Logic

First-order logic

- is expressive:
 - can be used to formalize mathematical concepts,
 - can be used to encode Turing machines,
 - but cannot axiomatize natural numbers or uncountable sets,
- has important decidable fragments,
- has interesting logical properties (model and proof theory).

First-order logic is also called (first-order) **predicate logic**.

3.1 Syntax

Syntax:

- nonlogical symbols (domain-specific)
⇒ terms, atomic formulas
- logical connectives (domain-independent)
⇒ Boolean combinations, quantifiers

Signatures

A signature $\Sigma = (\Omega, \Pi)$ fixes an alphabet of nonlogical symbols, where

- Ω is a set of **function symbols** f with **arity** $n \geq 0$,
written $\text{arity}(f) = n$,
- Π is a set of **predicate symbols** P with **arity** $m \geq 0$,
written $\text{arity}(P) = m$.

Function symbols are also called **operator symbols**.

If $n = 0$ then f is also called a **constant (symbol)**.

If $m = 0$ then P is also called a **propositional variable**.

Signatures

We will usually use

b, c, d for constant symbols,

f, g, h for nonconstant function symbols,

P, Q, R, S for predicate symbols.

Convention: We will usually write $f/n \in \Omega$ instead of $f \in \Omega$, $\text{arity}(f) = n$ (analogously for predicate symbols).

Signatures

Refined concept for practical applications:

many-sorted signatures (corresponds to simple type systems in programming languages);

no big change from a logical point of view.

Variables

Predicate logic admits the formulation of abstract, schematic assertions.
(Object) variables are the technical tool for schematization.

We assume that X is a given countably infinite set of symbols which we use to denote **variables**.

Terms

Terms over Σ and X (Σ -terms) are formed according to these syntactic rules:

$$\begin{array}{lcl} s, t, u, v & ::= & x \quad , x \in X \quad \text{(variable)} \\ & | & f(s_1, \dots, s_n) \quad , f/n \in \Omega \quad \text{(functional term)} \end{array}$$

By $T_\Sigma(X)$ we denote the set of Σ -terms (over X).

A term not containing any variable is called a **ground term**.

By T_Σ we denote the set of Σ -ground terms.

Atoms

Atoms (also called atomic formulas) over Σ are formed according to this syntax:

$$A, B ::= P(s_1, \dots, s_m) \text{ , } P/m \in \Pi \quad (\text{nonequational atom}) \\ \left[\mid (s \approx t) \quad (\text{equation}) \right]$$

Whenever we admit equations as atomic formulas we are in the realm of **first-order logic with equality**. Admitting equality does not really increase the expressiveness of first-order logic (see next part). But deductive systems where equality is treated specifically are much more efficient.

Literals

$L ::= A$ (positive literal)
 | $\neg A$ (negative literal)

Clauses

$C, D ::= \perp$ (empty clause)
 | $L_1 \vee \cdots \vee L_k, \ k \geq 1$ (nonempty clause)

General First-Order Formulas

$F_{\Sigma}(X)$ is the set of **first-order formulas** over Σ defined as follows:

F, G, H	$::=$	\perp	(falsum)
		\top	(verum)
		A	(atomic formula)
		$\neg F$	(negation)
		$(F \wedge G)$	(conjunction)
		$(F \vee G)$	(disjunction)
		$(F \rightarrow G)$	(implication)
		$(F \leftrightarrow G)$	(equivalence)
		$\forall x F$	(universal quantification)
		$\exists x F$	(existential quantification)

Notational Conventions

We omit parentheses according to the conventions for propositional logic.

$\forall x_1, \dots, x_n F$ and $\exists x_1, \dots, x_n F$ abbreviate

$\forall x_1 \dots \forall x_n F$ and $\exists x_1 \dots \exists x_n F$.

Notational Conventions

We use infix, prefix, postfix, or mixfix notation with the usual operator precedences.

Examples:

$$s + t * u \quad \text{for} \quad +(s, *(t, u))$$

$$s * u \leq t + v \quad \text{for} \quad \leq (*(s, u), +(t, v))$$

$$-s \quad \text{for} \quad -(s)$$

$$s! \quad \text{for} \quad !(s)$$

$$|s| \quad \text{for} \quad |-(s)|$$

$$0 \quad \text{for} \quad 0()$$

Example: Peano Arithmetic

$$\Sigma_{\text{PA}} = (\Omega_{\text{PA}}, \Pi_{\text{PA}})$$

$$\Omega_{\text{PA}} = \{0/0, +/2, */2, s/1\}$$

$$\Pi_{\text{PA}} = \{</2\}$$

Examples of formulas over this signature are

$$\forall x, y ((x < y \vee x \approx y) \leftrightarrow \exists z (x + z \approx y))$$

$$\exists x \forall y (x + y \approx y)$$

$$\forall x, y (x * s(y) \approx x * y + x)$$

$$\forall x, y (s(x) \approx s(y) \rightarrow x \approx y)$$

$$\forall x \exists y (x < y \wedge \neg \exists z (x < z \wedge z < y))$$

Positions in Terms and Formulas

The set of positions is extended from propositional logic to first-order logic:

The **positions** of a term s (formula F):

$$\text{pos}(x) = \{\varepsilon\},$$

$$\text{pos}(f(s_1, \dots, s_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^n \{i p \mid p \in \text{pos}(s_i)\},$$

$$\text{pos}(P(t_1, \dots, t_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^n \{i p \mid p \in \text{pos}(t_i)\},$$

$$\text{pos}(\forall x F) = \{\varepsilon\} \cup \{1 p \mid p \in \text{pos}(F)\},$$

$$\text{pos}(\exists x F) = \{\varepsilon\} \cup \{1 p \mid p \in \text{pos}(F)\}.$$

Positions in Terms and Formulas

The prefix order \leq , the subformula (subterm) operator, the formula (term) replacement operator, and the size operator are extended accordingly.

Variables

The **set of variables** occurring in a term t is denoted by $\text{var}(t)$ (and analogously for atoms, literals, clauses, and formulas).

Bound and Free Variables

In $Qx F$, $Q \in \{\exists, \forall\}$, we call F the **scope** of the quantifier Qx .

An *occurrence* of a variable x is called **bound** if it is inside the scope of a quantifier Qx .

Any other occurrence of a variable is called **free**.

Formulas without free variables are called **closed formulas** (or **sentential forms**).

Formulas without variables are called **ground**.

Bound and Free Variables

Example:

$$\begin{array}{c} \text{scope of } \forall y \\ \overbrace{\hspace{10em}} \\ \text{scope of } \forall x \\ \overbrace{\hspace{3em}} \\ \forall y \quad ((\forall x \quad \overbrace{P(x)} \quad) \rightarrow R(x, y)) \end{array}$$

The occurrence of y is bound, as is the first occurrence of x . The second occurrence of x is a free occurrence.

Substitutions

Substitution is a fundamental operation on terms and formulas that occurs in all inference systems for first-order logic.

Substitutions are mappings

$$\sigma : X \rightarrow T_{\Sigma}(X)$$

such that the **domain** of σ , that is, the set

$$\text{dom}(\sigma) = \{x \in X \mid \sigma(x) \neq x\},$$

is finite. The set of variables **introduced** by σ , that is, the set of variables occurring in one of the terms $\sigma(x)$, with $x \in \text{dom}(\sigma)$, is denoted by **codom**(σ).

Substitutions

Substitutions are often written as $\{x_1 \mapsto s_1, \dots, x_n \mapsto s_n\}$, with x_i pairwise distinct, and then denote the mapping

$$\{x_1 \mapsto s_1, \dots, x_n \mapsto s_n\}(y) = \begin{cases} s_i, & \text{if } y = x_i \\ y, & \text{otherwise} \end{cases}$$

We also write $x\sigma$ for $\sigma(x)$.

The **modification** of a substitution σ at x is defined as follows:

$$\sigma[x \mapsto t](y) = \begin{cases} t, & \text{if } y = x \\ \sigma(y), & \text{otherwise} \end{cases}$$

Why Substitution is Complicated

We define the application of a substitution σ to a term t or formula F by structural induction over the syntactic structure of t or F by the equations on the next slide.

In the presence of quantification it is surprisingly complex:

We must not only ensure that bound variables are not replaced by σ .

We must also make sure that the (free) variables in the codomain of σ are not *captured* upon placing them into the scope of a quantifier Qy .

Hence the bound variable must be renamed into a “fresh,” that is, previously unused, variable z .

Application of a Substitution

“Homomorphic” extension of σ to terms and formulas:

$$f(s_1, \dots, s_n)\sigma = f(s_1\sigma, \dots, s_n\sigma)$$

$$\perp\sigma = \perp$$

$$\top\sigma = \top$$

$$P(s_1, \dots, s_n)\sigma = P(s_1\sigma, \dots, s_n\sigma)$$

$$(u \approx v)\sigma = (u\sigma \approx v\sigma)$$

$$\neg F\sigma = \neg(F\sigma)$$

$$(F \circ G)\sigma = (F\sigma \circ G\sigma) \quad \text{for each binary connective } \circ$$

$$(Qx F)\sigma = Qz (F\sigma[x \mapsto z]) \quad \text{with } z \text{ a fresh variable}$$

Application of a Substitution

If $s = t\sigma$ for some substitution σ ,
we call the term s an **instance** of the term t ,
and we call t a **generalization** of s (analogously for formulas).

3.2 Semantics

To give semantics to a logical system means to define a notion of truth for the formulas. The concept of truth that we will now define for first-order logic goes back to Tarski.

As in the propositional case, we use a two-valued logic with truth values “true” and “false” denoted by 1 and 0, respectively.

Algebras

A Σ -algebra (also called Σ -interpretation or Σ -structure) is a triple

$$\mathcal{A} = (U_{\mathcal{A}}, (f_{\mathcal{A}} : U_{\mathcal{A}}^n \rightarrow U_{\mathcal{A}})_{f/n \in \Omega}, (P_{\mathcal{A}} \subseteq U_{\mathcal{A}}^m)_{P/m \in \Pi})$$

where $U_{\mathcal{A}} \neq \emptyset$ is a set, called the **universe** of \mathcal{A} .

By $\Sigma\text{-Alg}$ we denote the class of all Σ -algebras.

Σ -algebras generalize the valuations from propositional logic.

Assignments

A variable has no intrinsic meaning. The meaning of a variable has to be defined externally (explicitly or implicitly in a given context) by an assignment.

A (variable) assignment (over a given Σ -algebra \mathcal{A}) is a function $\beta : X \rightarrow U_{\mathcal{A}}$.

Variable assignments are the semantic counterparts of substitutions.

Value of a Term in \mathcal{A} with respect to β

By structural induction we define

$$\mathcal{A}(\beta) : T_{\Sigma}(X) \rightarrow U_{\mathcal{A}}$$

as follows:

$$\begin{aligned}\mathcal{A}(\beta)(x) &= \beta(x), & x \in X \\ \mathcal{A}(\beta)(f(s_1, \dots, s_n)) &= f_{\mathcal{A}}(\mathcal{A}(\beta)(s_1), \dots, \mathcal{A}(\beta)(s_n)), & f/n \in \Omega\end{aligned}$$

Value of a Term in \mathcal{A} with respect to β

In the scope of a quantifier we need to evaluate terms with respect to modified assignments. To that end, let $\beta[x \mapsto a] : X \rightarrow U_{\mathcal{A}}$, for $x \in X$ and $a \in U_{\mathcal{A}}$, denote the assignment

$$\beta[x \mapsto a](y) = \begin{cases} a & \text{if } x = y \\ \beta(y) & \text{otherwise} \end{cases}$$

Truth Value of a Formula in \mathcal{A} with respect to β

$\mathcal{A}(\beta) : F_{\Sigma}(X) \rightarrow \{0, 1\}$ is defined inductively as follows:

$$\mathcal{A}(\beta)(\perp) = 0$$

$$\mathcal{A}(\beta)(\top) = 1$$

$$\mathcal{A}(\beta)(P(s_1, \dots, s_n)) = \begin{array}{l} \text{if } (\mathcal{A}(\beta)(s_1), \dots, \mathcal{A}(\beta)(s_n)) \in P_{\mathcal{A}} \\ \text{then 1 else 0} \end{array}$$

$$\mathcal{A}(\beta)(s \approx t) = \begin{array}{l} \text{if } \mathcal{A}(\beta)(s) = \mathcal{A}(\beta)(t) \text{ then 1 else 0} \end{array}$$

Truth Value of a Formula in \mathcal{A} with respect to β

$\mathcal{A}(\beta) : F_{\Sigma}(X) \rightarrow \{0, 1\}$ is defined inductively as follows:

$$\mathcal{A}(\beta)(\neg F) = 1 - \mathcal{A}(\beta)(F)$$

$$\mathcal{A}(\beta)(F \wedge G) = \min(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

$$\mathcal{A}(\beta)(F \vee G) = \max(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

$$\mathcal{A}(\beta)(F \rightarrow G) = \max(1 - \mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

$$\mathcal{A}(\beta)(F \leftrightarrow G) = \text{if } \mathcal{A}(\beta)(F) = \mathcal{A}(\beta)(G) \text{ then } 1 \text{ else } 0$$

$$\mathcal{A}(\beta)(\forall x F) = \min_{a \in U_{\mathcal{A}}} \{ \mathcal{A}(\beta[x \mapsto a])(F) \}$$

$$\mathcal{A}(\beta)(\exists x F) = \max_{a \in U_{\mathcal{A}}} \{ \mathcal{A}(\beta[x \mapsto a])(F) \}$$

Example

The “standard” interpretation for Peano arithmetic:

$$U_{\mathbb{N}} = \{0, 1, 2, \dots\}$$

$$0_{\mathbb{N}} = 0$$

$$s_{\mathbb{N}} : n \mapsto n + 1$$

$$+_{\mathbb{N}} : (n, m) \mapsto n + m$$

$$*_{\mathbb{N}} : (n, m) \mapsto n * m$$

$$<_{\mathbb{N}} = \{(n, m) \mid n \text{ less than } m\}$$

Note that \mathbb{N} is just one out of many possible Σ_{PA} -interpretations.

Example

Values over \mathbb{N} for sample terms and formulas:

Under the assignment $\beta : x \mapsto 1, y \mapsto 3$ we obtain

$$\mathbb{N}(\beta)(s(x) + s(0)) = 3$$

$$\mathbb{N}(\beta)(x + y \approx s(y)) = 1$$

$$\mathbb{N}(\beta)(\forall x, y (x + y \approx y + x)) = 1$$

$$\mathbb{N}(\beta)(\forall z (z < y)) = 0$$

$$\mathbb{N}(\beta)(\forall x \exists y (x < y)) = 1$$

Ground Terms and Closed Formulas

If t is a ground term, then $\mathcal{A}(\beta)(t)$ does not depend on β , that is, $\mathcal{A}(\beta)(t) = \mathcal{A}(\beta')(t)$ for every β and β' .

Analogously, if F is a closed formula, then $\mathcal{A}(\beta)(F)$ does not depend on β , that is, $\mathcal{A}(\beta)(F) = \mathcal{A}(\beta')(F)$ for every β and β' .

Ground Terms and Closed Formulas

An element $a \in U_{\mathcal{A}}$ is called **term-generated** if $a = \mathcal{A}(\beta)(t)$ for some ground term t .

In general, not every element of an algebra is term-generated.

3.3 Models, Validity, and Satisfiability

F is **true** in \mathcal{A} under assignment β :

$$\mathcal{A}, \beta \models F \quad :\Leftrightarrow \quad \mathcal{A}(\beta)(F) = 1$$

F is **true** in \mathcal{A} (\mathcal{A} is a **model** of F ; F is **valid** in \mathcal{A}):

$$\mathcal{A} \models F \quad :\Leftrightarrow \quad \mathcal{A}, \beta \models F \quad \text{for all } \beta \in X \rightarrow U_{\mathcal{A}}$$

F is **valid** (or is a **tautology**):

$$\models F \quad :\Leftrightarrow \quad \mathcal{A} \models F \quad \text{for all } \mathcal{A} \in \Sigma\text{-Alg}$$

F is called **satisfiable** if there exist \mathcal{A} and β such that $\mathcal{A}, \beta \models F$.
Otherwise F is called **unsatisfiable**.

Entailment and Equivalence

F entails (implies) G (or G is a consequence of F), written $F \models G$, if for all $\mathcal{A} \in \Sigma\text{-Alg}$ and $\beta \in X \rightarrow U_{\mathcal{A}}$, we have

$$\mathcal{A}, \beta \models F \quad \Rightarrow \quad \mathcal{A}, \beta \models G$$

F and G are called equivalent, written $F \models\!\!\models G$, if for all $\mathcal{A} \in \Sigma\text{-Alg}$ and $\beta \in X \rightarrow U_{\mathcal{A}}$ we have

$$\mathcal{A}, \beta \models F \quad \Leftrightarrow \quad \mathcal{A}, \beta \models G$$

Entailment and Equivalence

Proposition 3.3.1:

$F \models G$ if and only if $F \rightarrow G$ is valid

Proposition 3.3.2:

$F \models\!\!\models G$ if and only if $F \leftrightarrow G$ is valid.

Extension to sets of formulas N as in propositional logic, e.g.:

$N \models F \quad :\Leftrightarrow \quad$ for all $\mathcal{A} \in \Sigma\text{-Alg}$ and $\beta \in X \rightarrow U_{\mathcal{A}}$:
if $\mathcal{A}, \beta \models G$ for all $G \in N$, then $\mathcal{A}, \beta \models F$.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 3.3.3:

Let F and G be formulas, let N be a set of formulas. Then

- (i) F is valid if and only if $\neg F$ is unsatisfiable.
- (ii) $F \models G$ if and only if $F \wedge \neg G$ is unsatisfiable.
- (iii) $N \models G$ if and only if $N \cup \{\neg G\}$ is unsatisfiable.

Hence in order to design a theorem prover (validity checker), it is sufficient to design a checker for unsatisfiability.

Substitution Lemma

Lemma 3.3.4:

Let \mathcal{A} be a Σ -algebra, let β be an assignment, let σ be a substitution. Then for any Σ -term t

$$\mathcal{A}(\beta)(t\sigma) = \mathcal{A}(\beta \circ \sigma)(t),$$

where $\beta \circ \sigma : X \rightarrow U_{\mathcal{A}}$ is the assignment $(\beta \circ \sigma)(x) = \mathcal{A}(\beta)(x\sigma)$.

Proposition 3.3.5:

Let \mathcal{A} be a Σ -algebra, let β be an assignment, let σ be a substitution. Then for every Σ -formula F

$$\mathcal{A}(\beta)(F\sigma) = \mathcal{A}(\beta \circ \sigma)(F).$$

Substitution Lemma

Corollary 3.3.6:

$$\mathcal{A}, \beta \models F\sigma \iff \mathcal{A}, \beta \circ \sigma \models F$$

These theorems basically express that the syntactic concept of substitution corresponds to the semantic concept of an assignment.

Two Lemmas

Lemma 3.3.7:

Let \mathcal{A} be a Σ -algebra. Let F be a Σ -formula with free variables x_1, \dots, x_n .

Then

$$\mathcal{A} \models \forall x_1, \dots, x_n F \text{ if and only if } \mathcal{A} \models F.$$

Two Lemmas

Lemma 3.3.8:

Let \mathcal{A} be a Σ -algebra.

Let F be a Σ -formula with free variables x_1, \dots, x_n .

Let σ be a substitution and let y_1, \dots, y_m be the free variables of $F\sigma$. Then

$$\mathcal{A} \models \forall x_1, \dots, x_n F \text{ implies } \mathcal{A} \models \forall y_1, \dots, y_m F\sigma.$$

3.4 Algorithmic Problems

Validity(F): $\models F$?

Satisfiability(F): F satisfiable?

Entailment(F, G): does F entail G ?

Model(\mathcal{A}, F): $\mathcal{A} \models F$?

Solve(\mathcal{A}, F): find an assignment β such that $\mathcal{A}, \beta \models F$.

Solve(F): find a substitution σ such that $\models F\sigma$.

Abduce(F): find G with “certain properties” such that $G \models F$.

Theory of an Algebra

Let $\mathcal{A} \in \Sigma\text{-Alg}$. The (first-order) theory of \mathcal{A} is defined as

$$\text{Th}(\mathcal{A}) = \{G \in F_{\Sigma}(X) \mid \mathcal{A} \models G\}$$

Problem of axiomatizability:

Given an algebra \mathcal{A} (or a class of algebras) can one axiomatize $\text{Th}(\mathcal{A})$, that is, can one write down a formula F (or a semidecidable set F of formulas) such that

$$\text{Th}(\mathcal{A}) = \{G \mid F \models G\}?$$

Two Interesting Theories

Let $\Sigma_{\text{Pres}} = (\{0/0, s/1, +/2\}, \{<\})$ and $\mathbb{N}_+ = (\mathbb{N}, 0, s, +, <)$ its standard interpretation on the natural numbers.

$\text{Th}(\mathbb{N}_+)$ is called **Presburger arithmetic** (M. Presburger, 1929).

(There is no essential difference when one, instead of \mathbb{N} , considers the integer numbers \mathbb{Z} as standard interpretation.)

Presburger arithmetic is decidable in 3EXPTIME (D. Oppen, JCSS, 16(3):323–332, 1978), and in 2EXPSPACE, using automata-theoretic methods (and there is a constant $c \geq 0$ such that $\text{Th}(\mathbb{Z}_+) \notin \text{NTIME}(2^{2^{cn}})$).

Two Interesting Theories

However, $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *, <)$, the standard interpretation of $\Sigma_{PA} = (\{0/0, s/1, +/2, */2\}, \{<\})$, has as theory the so-called **Peano arithmetic** which is undecidable and not even semidecidable.

(Non)computability Results

1. For most signatures Σ , validity is undecidable for Σ -formulas.
(One can easily encode Turing machines in most signatures.)
2. Gödel's completeness theorem:
For each signature Σ , the set of valid Σ -formulas is semidecidable.
(We will prove this by giving complete deduction systems.)
3. Gödel's incompleteness theorem:
For $\Sigma = \Sigma_{PA}$ and $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *, <)$, the theory $\text{Th}(\mathbb{N}_*)$ is not semidecidable.

These complexity results motivate the study of subclasses of formulas (**fragments**) of first-order logic.