

AAA528: Computational Logic

Lecture 4 — First-Order Logic

Hakjoo Oh
2026 Spring

First-Order Logic

- An extension of propositional logic with predicates, functions, and quantifiers.
- First-order logic is also called predicate logic, first-order predicate calculus, and relational logic.
- First-order logic is expressive enough to reason about programs.
- However, completely automated reasoning is not possible.

Terms (Variables, Constants, and Functions)

- Terms denote the objects that we are reasoning about.
- While formulas in PL evaluate to true or false, terms in FOL evaluate to values in an underlying domain such as integers, strings, lists, etc.
- Terms in FOL are defined by the grammar:

$$t \rightarrow x \mid c \mid f(t_1, \dots, t_n)$$

- ▶ Basic terms are **variables** (x, y, z, \dots) and **constants** (a, b, c, \dots).
- ▶ Composite terms include n -ary **functions** applied to n terms, i.e., $f(t_1, \dots, t_n)$, where t_i s are terms.
 - ★ A constant can be viewed as a 0-ary function.
- Examples:
 - ▶ $f(a)$, a unary function f applied to a constant
 - ▶ $g(x, b)$, a binary function g applied to a variable x and a constant b
 - ▶ $f(g(x, f(b)))$

Predicates

- The propositional variables of PL are generalized to **predicates** in FOL, denoted p, q, r, \dots
- An n -ary predicate takes n terms as arguments.
- A FOL propositional variable is a 0-ary predicate, denoted P, Q, \dots
- Examples:
 - ▶ P , a propositional variable (or 0-ary predicate)
 - ▶ $p(f(x), g(x, f(x)))$, a binary predicate applied to two terms

Syntax

- **Atom**: basic elements
 - ▶ truth symbols \perp (“false”) and \top (“true”)
 - ▶ n -ary predicates applied to n terms
- **Literal**: an atom α or its negation $\neg\alpha$.
- **Formula**: a literal or application of a logical connective to formulas, or the application of a quantifier to a formula.

F	\rightarrow	\perp		\top		$p(t_1, \dots, t_n)$	atom
		$\neg F$					negation (“not”)
		$F_1 \wedge F_2$					conjunction (“and”)
		$F_1 \vee F_2$					disjunction (“or”)
		$F_1 \rightarrow F_2$					implication (“implies”)
		$F_1 \leftrightarrow F_2$					iff (“if and only if”)
		$\exists x.F[x]$					existential quantification
		$\forall x.F[x]$					universal quantification

Notations on Quantification

- In $\forall x.F[x]$ and $\exists x.F[x]$, x is the **quantified variable** and $F[x]$ is the **scope** of the quantifier. We say x is **bound** in $F[x]$.
- $\forall x.\forall y.F[x, y]$ is often abbreviated by $\forall x, y.F[x, y]$.
- The scope of the quantified variable extends as far as possible: e.g.,

$$\forall x.p(f(x), x) \rightarrow (\exists y.p(f(g(x, y)), g(x, y))) \wedge q(x, f(x))$$

- A variable is **free** in $F[x]$ if it is not bound. $\mathbf{free}(F)$ and $\mathbf{bound}(F)$ denote the free and bound variables of F , respectively. A formula F is **closed** if F has no free variables. E.g.,

$$\forall x.p(f(x), y) \rightarrow \forall y.p(f(x), y)$$

- If $\mathbf{free}(F) = \{x_1, \dots, x_n\}$, then its **universal closure** is $\forall x_1 \dots \forall x_n.F$ and its **existential closure** is $\exists x_1 \dots \exists x_n.F$. They are usually written $\forall * .F$ and $\exists * .F$.

Example FOL Formulas

- Every dog has its day.

$$\forall x. \mathit{dog}(x) \rightarrow \exists y. \mathit{day}(y) \wedge \mathit{itsDay}(x, y)$$

- Some dogs have more days than others.

$$\exists x, y. \mathit{dog}(x) \wedge \mathit{dog}(y) \wedge \#days(x) > \#days(y)$$

- The length of one side of a triangle is less than the sum of the lengths of the other two sides.

$$\forall x, y, z. \mathit{triangle}(x, y, z) \rightarrow \mathit{length}(x) < \mathit{length}(y) + \mathit{length}(z)$$

- Fermat's Last Theorem.

$$\forall n. \mathit{integer}(n) \wedge n > 2$$

$$\rightarrow \forall x, y, z.$$

$$\mathit{integer}(x) \wedge \mathit{integer}(y) \wedge \mathit{integer}(z) \wedge x > 0 \wedge y > 0 \wedge z > 0$$

$$\rightarrow x^n + y^n \neq z^n$$

Interpretation

A FOL **interpretation** $I : (D_I, \alpha_I)$ is a pair of a domain and an assignment.

- D_I is a nonempty set of values such as integers, real numbers, etc.
- α_I maps variables, constant, functions, and predicate symbols to elements, functions, and predicates over D_I .
 - ▶ each variable x is assigned a value from D_I
 - ▶ each n -ary function symbol f is assigned an n -ary function $f_I : D_I^n \rightarrow D_I$.
 - ▶ each n -ary predicate symbol p is assigned an n -ary predicate $p_I : D_I^n \rightarrow \{\text{true}, \text{false}\}$.
- Arbitrary terms and atoms are evaluated recursively:

$$\begin{aligned}\alpha_I[f(t_1, \dots, t_n)] &= \alpha_I[f](\alpha_I[t_1], \dots, \alpha_I[t_n]) \\ \alpha_I[p(t_1, \dots, t_n)] &= \alpha_I[p](\alpha_I[t_1], \dots, \alpha_I[t_n])\end{aligned}$$

Example

$$F : x + y > z \rightarrow y > z - x$$

- Note $+$, $-$, $>$ are just symbols: we could have written

$$p(f(x, y), z) \rightarrow p(y, g(z, x)).$$

- Domain: $D_I = \mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$
- Assignment:

$$\alpha_I = \{+ \mapsto +_{\mathbb{Z}}, - \mapsto -_{\mathbb{Z}}, > \mapsto >_{\mathbb{Z}}, x \mapsto 13, y \mapsto 42, z \mapsto 1, \dots\}$$

Semantics of First-Order Logic

Given an interpretation $I : (D_I, \alpha_I)$, $I \models F$ or $I \not\models F$.

$$I \models \top, \quad I \not\models \perp,$$

$$I \models p(t_1, \dots, t_n) \quad \text{iff} \quad \alpha_I[p(t_1, \dots, t_n)] = \mathbf{true}$$

$$I \models \neg F \quad \text{iff} \quad I \not\models F$$

$$I \models F_1 \wedge F_2 \quad \text{iff} \quad I \models F_1 \text{ and } I \models F_2$$

$$I \models F_1 \vee F_2 \quad \text{iff} \quad I \models F_1 \text{ or } I \models F_2$$

$$I \models F_1 \rightarrow F_2 \quad \text{iff} \quad I \not\models F_1 \text{ or } I \models F_2$$

$$I \models F_1 \leftrightarrow F_2 \quad \text{iff} \quad (I \models F_1 \text{ and } I \models F_2) \text{ or } (I \not\models F_1 \text{ and } I \not\models F_2)$$

$$I \models \forall x.F \quad \text{iff for all } v \in D_I, I \triangleleft \{x \mapsto v\} \models F$$

$$I \models \exists x.F \quad \text{iff there exists } v \in D_I, I \triangleleft \{x \mapsto v\} \models F$$

where $J : I \triangleleft \{x \mapsto v\}$ denotes an x -variant of I :

- $D_J = D_I$
- $\alpha_J[\mathbf{y}] = \alpha_I[\mathbf{y}]$ for all constant, free variable, function, and predicate symbols \mathbf{y} , except that $\alpha_J(x) = v$.

Example

Consider the formula:

$$F : \exists x. f(x) = g(x)$$

and the interpretation $I : (D : \{v_1, v_2\}, \alpha_I)$:

$$\alpha_I : \{f(v_1) \mapsto v_1, f(v_2) \mapsto v_2, g(v_1) \mapsto v_2, g(v_2) \mapsto v_1\}$$

Compute the truth value of F under I as follows:

1. $I \triangleleft \{x \mapsto v\} \not\models f(x) = g(x)$ for $v \in D$
2. $I \not\models \exists x. f(x) = g(x)$ since $v \in D$ is arbitrary

Satisfiability and Validity

- A formula F is *satisfiable* iff there exists an interpretation I such that $I \models F$.
- A formula F is *valid* iff for all interpretations I , $I \models F$.
- Technically, satisfiability and validity are defined for closed FOL formulas. Convention for formulas with free variables:
 - ▶ If we say that a formula F such that $\mathbf{free}(F) \neq \emptyset$ is valid, we mean that its universal closure $\forall * .F$ is valid.
 - ▶ If we say that F is satisfiable, we mean that its existential closure $\exists * .F$ is satisfiable.
 - ▶ Duality still holds:

$$\forall * .F \text{ is valid} \iff \exists * .\neg F \text{ is unsatisfiable.}$$

Extension of the Semantic Argument Method

Most of the proof rules from PL carry over to FOL:

$$\frac{I \models \neg F}{I \not\models F}$$

$$\frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{I \models F, I \models G}$$

$$\frac{I \not\models F \wedge G}{I \not\models F \mid I \not\models G}$$

$$\frac{I \models F \vee G}{I \models F \mid I \models G}$$

$$\frac{I \not\models F \vee G}{I \not\models F, I \not\models G}$$

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G}$$

$$\frac{I \not\models F \rightarrow G}{I \models F, I \not\models G}$$

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \models \neg F \wedge \neg G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

Rules for Quantifiers

“Universal” rules:

- Universal elimination I:

$$\frac{I \models \forall x.F}{I \triangleleft \{x \mapsto v\} \models F} \text{ for any } v \in D_I$$

- Existential elimination I:

$$\frac{I \not\models \exists x.F}{I \triangleleft \{x \mapsto v\} \not\models F} \text{ for any } v \in D_I$$

These rules are usually applied using a domain element v that was introduced earlier in the proof.

Rules for Quantifiers

“Existential” rules:

- Existential elimination II:

$$\frac{I \models \exists x.F}{I \triangleleft \{x \mapsto v\} \models F} \text{ for a fresh } v \in D_I$$

- Universal elimination II:

$$\frac{I \not\models \forall x.F}{I \triangleleft \{x \mapsto v\} \not\models F} \text{ for a fresh } v \in D_I$$

When applying these rules, v must not have been previously used in the proof.

Contradiction Rule

A contradiction exists if two variants of the original interpretation I disagree on the truth value of an n -ary predicate p for a given tuple of domain values:

$$\frac{\begin{array}{l} J : I \triangleleft \dots \models p(s_1, \dots, s_n) \\ K : I \triangleleft \dots \not\models p(t_1, \dots, t_n) \end{array}}{I \models \perp} \quad \text{for } i \in \{1, \dots, n\}, \alpha_J[s_i] = \alpha_K[t_i]$$

Example 1

Prove that the formula is valid:

$$F : (\forall x.p(x)) \rightarrow (\forall y.p(y))$$

Suppose not; there is an interpretation I such that $I \not\models F$.

1. $I \not\models F$ assumption
2. $I \models \forall x.p(x)$ 1 and \rightarrow
3. $I \not\models \forall y.p(y)$ 1 and \rightarrow
4. $I \triangleleft \{y \mapsto v\} \not\models p(y)$ 3 and \forall , for some $v \in D_I$
5. $I \triangleleft \{x \mapsto v\} \models p(x)$ 2 and \forall
6. $I \models \perp$ 4 and 5

Example 2

Prove that the formula is valid:

$$F : (\forall x.p(x)) \leftrightarrow (\neg\exists x.\neg p(x))$$

We need to show both of forward and backward directions.

$$F_1 : (\forall x.p(x)) \rightarrow (\neg\exists x.\neg p(x)), \quad F_2 : (\forall x.p(x)) \leftarrow (\neg\exists x.\neg p(x))$$

Suppose F_1 is not valid; there is an interpretation I such that $I \not\models F_1$.

1. $I \models \forall x.p(x)$ assumption
2. $I \not\models \neg\exists x.\neg p(x)$ assumption
3. $I \models \exists x.\neg p(x)$ 2 and \neg
4. $I \triangleleft \{x \mapsto v\} \models \neg p(x)$ 3 and \exists , for some $v \in D_I$
5. $I \triangleleft \{x \mapsto v\} \models p(x)$ 1 and \forall
6. $I \models \perp$ 4 and 5

Exercise) Prove that F_2 is valid.

Example 3

Prove that the formula is valid:

$$F : p(a) \rightarrow \exists x.p(x).$$

Assume F is invalid and derive a contradiction:

1. $I \not\models F$ assumption
2. $I \models p(a)$ 1 and \rightarrow
3. $I \not\models \exists x.p(x)$ 1 and \rightarrow
4. $I \triangleleft \{x \mapsto \alpha_I[a]\} \not\models p(x)$ 3 and \exists
5. $I \models \perp$ 2, 4

Example 4

Prove that the formula is invalid:

$$F : (\forall x.p(x, x)) \rightarrow (\exists x.\forall y.p(x, y))$$

It suffices to find an interpretation I such that $I \models \neg F$. Choose $D_I = \{0, 1\}$ and $p_I = \{(0, 0), (1, 1)\}$. The interpretation falsifies F .

Soundness and Completeness of FOL

A proof system is **sound** if every provable formula is valid. It is **complete** if every valid formula is provable.

Theorem (Sound)

If every branch of a semantic argument proof of $\mathbf{I} \not\models \mathbf{F}$ closes, then \mathbf{F} is valid.

Theorem (Complete)

Each valid formula \mathbf{F} has a semantic argument proof.

Substitution

- A substitution is a map from FOL formulas to FOL formulas:

$$\sigma : \{F_1 \mapsto G_1, \dots, F_n \mapsto G_n\}$$

- To compute $F\sigma$, replace each occurrence of F_i in F by G_i simultaneously.
- For example, consider formula

$$F : (\forall x.p(x, y)) \rightarrow q(f(y), x)$$

and substitution

$$\sigma : \{x \mapsto g(x), y \mapsto f(x), q(f(y), x) \mapsto \exists x.h(x, y)\}$$

Then,

$$F\sigma : (\forall x.p(g(x), f(x))) \rightarrow \exists x.h(x, y)$$

Safe Substitution

- A restricted application of substitution, which has a useful semantic property.
- Idea: Before applying substitution, replace bound variables to fresh variables.
- For example, consider formula

$$F : (\forall x.p(x, y)) \rightarrow q(f(y), x)$$

and substitution

$$\sigma : \{x \mapsto g(x), y \mapsto f(x), q(f(y), x) \mapsto \exists x.h(x, y)\}$$

Then, safe substitution proceeds

- 1 Renaming: $(\forall x'.p(x', y)) \rightarrow q(f(y), x)$
- 2 Substitution: $(\forall x'.p(x', f(x))) \rightarrow \exists x.h(x, y)$

Safe Substitution

A FOL version of Substitution of Equivalent Formulas:

Theorem

Consider substitution

$$\sigma : \{F_1 \mapsto G_1, \dots, G_n \mapsto G_n\}$$

such that for each i , $F_i \iff G_i$. Then $F \iff F\sigma$ when $F\sigma$ is computed as a safe substitution.

A FOL version of Valid Templates:

Theorem

If H is a valid formula schema and σ is a substitution obeying H 's side conditions, then $H\sigma$ is also valid.

Examples on Valid Templates

- Consider valid formula schema:

$$H : (\forall x.F) \leftrightarrow (\neg \exists x.\neg F)$$

The formula

$$G : (\forall x.\exists y.q(x, y)) \leftrightarrow (\neg \exists x.\neg \exists y.q(x, y))$$

is valid because $G = H\sigma$ for $\sigma : \{F \mapsto \exists y.q(x, y)\}$.

- Consider valid formula schema:

$$H : (\forall x.F) \leftrightarrow F \quad \text{provided } x \notin \mathbf{free}(F)$$

The formula

$$G : (\forall x.\exists y.p(z, y)) \leftrightarrow \exists y.p(z, y)$$

is valid because $G = H\sigma$ for $\sigma : \{F \mapsto \exists y.p(z, y)\}$.

Negation Normal Form

- A FOL formula F can be transformed into NNF by using the following equivalences:

$$\begin{array}{lll} \neg\neg F_1 & \iff & F_1 \\ \neg\top & \iff & \perp \\ \neg\perp & \iff & \top \\ \neg(F_1 \wedge F_2) & \iff & \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) & \iff & \neg F_1 \wedge \neg F_2 \\ F_1 \rightarrow F_2 & \iff & \neg F_1 \vee F_2 \\ F_1 \leftrightarrow F_2 & \iff & (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1) \\ \neg\forall x.F[x] & \iff & \exists x.\neg F[x] \\ \neg\exists x.F[x] & \iff & \forall x.\neg F[x] \end{array}$$

Example

Convert the formula into NNF:

$$G : \forall x. (\exists y. p(x, y) \wedge p(x, z)) \rightarrow \exists w. p(x, w)$$

- ① Use the equivalence $F_1 \rightarrow F_2 \iff \neg F_1 \vee F_2$:

$$\forall x. \neg(\exists y. p(x, y) \wedge p(x, z)) \vee \exists w. p(x, w)$$

- ② Use the equivalence $\neg \exists x. F[x] \iff \forall x. \neg F[x]$:

$$\forall x. (\forall y. \neg(p(x, y) \wedge p(x, z))) \vee \exists w. p(x, w)$$

- ③ Use De Morgan's Law:

$$\forall x. (\forall y. \neg p(x, y) \vee \neg p(x, z)) \vee \exists w. p(x, w)$$

Prenex Normal Form (PNF)

- A formula is in **prenex normal form (PNF)** if all of its quantifiers appear at the beginning of the formula:

$$\mathbf{Q}_1 x_1 \dots \mathbf{Q}_n x_n . F[x_1, \dots, x_n]$$

where $\mathbf{Q}_i \in \{\forall, \exists\}$ and F is quantifier-free.

- Every FOL F has an equivalent PNF. To convert F into PNF,
 - 1 Convert F into NNF: F_1
 - 2 Rename quantified variables to unique names: F_2
 - 3 Remove all quantifiers from F_2 : F_3
 - 4 Add the quantifiers before F_3 :

$$F_4 : \mathbf{Q}_1 x_1 \dots \mathbf{Q}_n x_n . F_3$$

where \mathbf{Q}_i are the quantifiers such that if \mathbf{Q}_j is in the scope of \mathbf{Q}_i in F_1 , then $i < j$.

- A FOL formula is in CNF (DNF) if it is in PNF and its main quantifier-free subformula is in CNF (DNF).

Example

$$F : \forall x. \neg(\exists y. p(x, y) \wedge p(x, z)) \vee \exists y. p(x, y)$$

- ① Conversion to NNF:

$$F_1 : \forall x. (\forall y. \neg p(x, y) \vee \neg p(x, z)) \vee \exists y. p(x, y)$$

- ② Rename quantified variables:

$$F_2 : \forall x. (\forall y. \neg p(x, y) \vee \neg p(x, z)) \vee \exists w. p(x, w)$$

- ③ Remove all quantifiers:

$$F_3 : \neg p(x, y) \vee \neg p(x, z) \vee p(x, w)$$

- ④ Add the quantifiers before F_3 :

$$F_4 : \forall x. \forall y. \exists w. \neg p(x, y) \vee \neg p(x, z) \vee p(x, w)$$

Note that $\forall x. \exists w. \forall y. F_3$ is okay, but $\forall y. \exists w. \forall x. F_3$ is not.

Decidability

- Satisfiability can be formalized as a decision problem in formal languages.
- Ex) Let L_{PL} be the set of all satisfiable formulas. Given w , is $w \in L_{PL}$?
- A formal language L is decidable if there exists a procedure that, given a word w , (1) eventually halts and (2) answer yes if $w \in L$ and no if $w \notin L$. Otherwise, L is undecidable.
- L_{PL} is decidable but L_{FOL} is not.

Summary

- Syntax and semantics of first-order logic
- Satisfiability and validity
- Substitution, Normal forms