COSE419: Software Verification

Lecture 6 — First-Order Logic

Hakjoo Oh
2024 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:

$$t \to x \mid c \mid f(t_1, \ldots, f_n)$$

  - Basic terms are **variables** (x, y, z, ...) and **constants** (a, b, c, ...).
  - Composite terms include n-ary **functions** applied to n terms, i.e., $f(t_1, \ldots, t_n)$, where t_is 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, ...
- An n-ary predicate takes n terms as arguments.
- A FOL propositional variable is a 0-ary predicate, denoted $P, Q, \ldots$
- 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 $\bot$ ("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.

$$
\begin{array}{rcll}
F & \to & \bot \mid \top \mid p(t_1, \ldots, t_n) & \text{atom} \\
 & \mid & \neg F & \text{negation ("not")} \\
 & \mid & F_1 \land F_2 & \text{conjunction ("and")} \\
 & \mid & F_1 \lor F_2 & \text{disjunction ("or")} \\
 & \mid & F_1 \to F_2 & \text{implication ("implies")} \\
 & \mid & F_1 \leftrightarrow F_2 & \text{iff ("if and only if")} \\
 & \mid & \exists x. F[x] & \text{existential quantification} \\
 & \mid & \forall x. F[x] & \text{universal quantification}
\end{array}
$$

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

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

- A variable is **free** in $F[x]$ if it is not bound. **free**(F) and **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 **free**$(F) = \{x_1, \ldots, x_n\}$, then its **universal closure** is $\forall x_1 \ldots \forall x_n.F$ and its **existential closure** is $\exists x_1 \ldots \exists x_n.F$. They are usually written $\forall * .F$ and $\exists * .F$.

Example FOL Formulas

- Every cat has its day.

$$\forall x. cat(x) \rightarrow \exists y. day(y) \land itsDay(x, y)$$

- Some cats have more days than others.

$$\exists x, y. cat(x) \land cat(y) \land \#days(x) > \#days(y)$$

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

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

- Fermat's Last Theorem.

$$\begin{array}{l} \forall n. integer(n) \land n > 2 \\ \quad \rightarrow \forall x, y, z. \\ \qquad integer(x) \land integer(y) \land integer(z) \land x > 0 \land y > 0 \land z > 0 \\ \qquad\quad \rightarrow x^n + y^n \neq z^n \end{array}$$

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 $f_I : D_I^n \to D_I$.
  - each n-ary predicate symbol p is assigned an n-ary predicate $p_I : D_I^n \to \{\textbf{true}, \textbf{false}\}$.
- Arbitrary terms and atoms are evaluated recursively:

$$\begin{array}{rcl} \alpha_I[f(t_1, \ldots, t_n)] & = & \alpha_I[f](\alpha_I[t_1], \ldots, \alpha_I[t_n]) \\ \alpha_I[p(t_1, \ldots, t_n)] & = & \alpha_I[p](\alpha_I[t_1], \ldots, \alpha_I[t_n]) \end{array}$$

Example

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

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

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

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

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

Semantics of First-Order Logic

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

$$
\begin{array}{ll}
I \models \top, \quad I \not\models \bot, & \\
I \models p(t_1, \ldots, t_n) & \text{iff} \;\; \alpha_I[p(t_1, \ldots, t_n)] = \textbf{true} \\
I \models \neg F & \text{iff} \;\; I \not\models F \\
I \models F_1 \wedge F_2 & \text{iff} \;\; I \models F_1 \text{ and } I \models F_2 \\
I \models F_1 \vee F_2 & \text{iff} \;\; I \models F_1 \text{ or } I \models F_2 \\
I \models F_1 \rightarrow F_2 & \text{iff} \;\; I \not\models F_1 \text{ or } I \models F_2 \\
I \models F_1 \leftrightarrow F_2 & \text{iff} \;\; (I \models F_1 \text{ and } I \models F_2) \text{ or } (I \not\models F_1 \text{ and } I \not\models F_2) \\
I \models \forall x.F & \text{iff for all } v \in D_I, I \lhd \{x \mapsto v\} \models F \\
I \models \exists x.F & \text{iff there exists } v \in D_I, I \lhd \{x \mapsto v\} \models F
\end{array}
$$

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

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

Example 1

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

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

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

1.	I	$\models$	$x + y > z$	since $\alpha_I[x + y > z] = 13 +_{\mathbb{Z}} 42 >_{\mathbb{Z}} 1$
2.	I	$\models$	$y > z - x$	since $\alpha_I[y > z - x] = 42 >_{\mathbb{Z}} 1 -_{\mathbb{Z}} 13$
3.	I	$\models$	F	by 1, 2, and the semantics of $\rightarrow$

Example 2

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:

$$\begin{array}{llcll} \textbf{1.} & I \lhd \{x \mapsto v\} & \not\models & f(x) = g(x) & \text{for all } v \in D \\ \textbf{2.} & I & \not\models & \exists x. f(x) = g(x) & \text{by the semantics of } \exists \end{array}$$

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:

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

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

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

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

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

Rules for Quantifiers

"Universal" rules:

- Universal elimination I:

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

- Existential elimination I:

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

There 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:

$$\frac{\begin{array}{c} J : I \lhd \cdots \models p(s_1, \ldots, s_n) \\ K : I \lhd \cdots \not\models p(t_1, \ldots, t_n) \end{array}}{I \models \bot} \text{ for } i \in \{1, \ldots, n\}, \alpha_J[s_i] = \alpha_K[t_i]$$

Example 1

Prove that the formula is valid:

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

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 $\to$
3.	$I \not\models \forall y. p(y)$	1 and $\to$
4.	$I \lhd \{y \mapsto v\} \not\models p(y)$	3 and $\forall$, for some $v \in D_I$
5.	$I \lhd \{x \mapsto v\} \models p(x)$	2 and $\forall$
6.	$I \models \bot$	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.

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

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.	$I \lhd \{x \mapsto v\} \models \neg p(x)$	3 and $\exists$, for some $v \in D_I$
5.	$I \lhd \{x \mapsto v\} \models p(x)$	1 and $\forall$
6.	$I \models \bot$	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.	$I \lhd \{x \mapsto \alpha_I[a]\} \not\models p(x)$	3 and $\exists$
5.	$I \models \bot$	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 $I \vDash \neg F$. 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 $I \not\models F$ closes, then F is valid.

Theorem (Complete)

Each valid formula F has a semantic argument proof.

Substitution

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

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

- 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

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

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: $(\forall x'.p(x', y)) \to q(f(y), x)$
2. Substitution: $(\forall x'.p(x', f(x))) \to \exists x.h(x, y)$

Safe Substitution

Proposition (Substitution of Equivalent Formulas)

Consider substitution

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

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

Formula Schema and Schema Substitution

- The relation $\forall x.\ p(x) \iff \neg\exists x.\ \neg p(x)$ is interesting but not general. Instead, we can prove the validity of **formula schema**

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

- A formula schema H contains at least one placeholder $F_1, F_2, \ldots$ and can also contain **side conditions** that specify that certain variables do not occur free in the placeholders, e.g.,

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

- Consider a substitution σ mapping placeholders to FOL formulas. A **schema substitution** is an (unrestricted) application of σ to a formula schema. It is legal only if σ obeys the side conditions of the formula schema.

Proposition (Formula Schema)

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

Examples

- 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:

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

  The formula

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

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

Proving Formula Schemata

Prove the validity of formula schema:

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

$(\rightarrow)$

$$\begin{array}{lll}
1. & I \models \forall x.F & \text{assumption} \\
2. & I \not\models F & \text{assumption} \\
3. & I \models F & 1, \forall, \text{since } x \notin \textbf{free}(F) \\
4. & I \models \bot & 2, 3
\end{array}$$

$(\leftarrow)$

$$\begin{array}{lll}
1. & I \not\models \forall x.F & \text{assumption} \\
2. & I \models F & \text{assumption} \\
3. & I \models \exists x.\neg F & 1, \neg \\
4. & I \models \neg F & 3, \exists, \text{since } x \notin \textbf{free}(F) \\
5. & I \models \bot & 2, 4
\end{array}$$

Negation Normal Form

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

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

Example

Convert the formula into NNF:

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

1. Use the equivalence $F_1 \rightarrow F_2 \iff \neg F_1 \lor F_2$:

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

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

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

3. Use De Morgan's Law:

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

Prenex Normal Form (PNF)

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

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

  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:

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

     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
- Soundness, completeness, decidability