EC4219: Software Engineering

Lecture 5 — First-Order Logic

Sunbeom So
2024 Spring

First-Order Logic (FOL)

- An extension of propositional logic (PL) with predicates, functions, and quantifiers.
- FOL is also called predicate logic, and the first-order predicate calculus.
- FOL is expressive enough to reason about programs.
- While the validity of PL formulas is decidable, the validity of FOL formulas is not.

Terms (Variables, Constants, and Functions)

- Terms are the objects that we are reasoning about.
- Terms in FOL evaluate to values other than truth values, such as integers, strings, or lists.
- Terms in FOL are defined by the grammar below:

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

  - Basic terms are variables (denoted x) and constants (denoted c).
  - Composite terms are functions. When a function takes n terms as arguments, we say that the function is an n-ary function (or, the function has the arity n).
    cf) A constant can be viewed as a 0-ary function.
- (Example) $g(x, b)$: a binary function g applied to a variable x and a constant b

Predicates

- The propositional variables of PL are generalized to predicates in FOL.
- An n-ary predicate takes n terms as arguments.
- A FOL propositional variable is a 0-ary predicate.
- For example, $p(f(x), g(x, f(x)))$ is a binary predicate applied to two terms.

Syntax

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

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

Notations: 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 $\forall x$. We say x is bound in $F[x]$.
- $\forall x.\forall y.F[x, y]$ can be abbreviated by $\forall x, y.F[x, y]$.
- The scope of the quantified variable extends as far as possible. For example, consider

$$\forall x.\, \overbrace{p(f(x), x) \to (\exists y.\, \underbrace{p(f(g(x, y)), g(x, y))}_{G}) \land q(x, f(x))}^{F}\, .$$

The scope of x is F, and the scope of y is G.

Notations: Quantification (cont'd)

- Given $F[x]$, a variable x is *free* if there is an occurence of x not bound by any quantifier.
- $\mathbf{free}(F)$ and $\mathbf{bound}(F)$ denote the free and bounbd variables of F, respectively.
- It is possible that $\mathbf{free}(F) \cap \mathbf{bound}(F) \neq \emptyset$.
  - Given $F : \forall x.p(f(x), y) \to \forall y.p(f(x), y)$, $\mathbf{free}(F) = \{y\}$ and $\mathbf{bound}(F) = \{x, y\}$.
- A formula F is closed if F has no free variables.
- Suppose $\mathbf{free}(F) = \{x_1, \cdots, x_n\}$. Then,
  - F's *universal closure* is $\forall x_1 \cdots \forall x_n.F$. Can be written $\forall * .F$.
  - F's *existential closure* is $\exists x_1 \cdots \exists x_n.F$. Can be written $\exists * .F$.

Interpretation

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

- A **domain** D_I is a nonempty set of values, such as integers or real numbers.
- An **assignment** α_I maps variables to elements of D_I. It also maps constants, function symbols, and predicate symbols to elements, functions, and predicates over D_I.
  - Each variable symbol x is assigned a value x_I from D_I.
  - Each constant 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 $p_I : D_I^n \to \{true, false\}$.

Example: Interpreation

Consider the formula

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

that contains the binary function symbols $+$ and $-$, and the binary predicate symbol $>$, and the variables x, y, and z.

- Each symbol is just a syntactical element. Their meaning is defined by the interpretation $I = (D_I, \alpha_I)$.
- Assume the domain is the integers: $D_I = \mathbb{Z} = \{\cdots, -1, 0, 1, \cdots\}$.
- Then, we may have the assignment

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

Semantics

- Semantics of FOL formulas are inductively defined as in PL.
- The cases with logical connectives ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) are handled in the same way as in PL.
- The semantics of predicates and quantifiers are new.

$\boxed{\text{Base cases}}$

$$I \models \top$$
$$I \not\models \bot$$
$$I \models p(t_1, \cdots, t_n) \quad \text{iff} \quad \alpha_I[p(t_1, \cdots, t_n)] = \mathit{true}$$

$\boxed{\text{Inductive cases}}$

$$
\begin{array}{lll}
I \models \neg F & \text{iff} & I \not\models F \\
I \models F_1 \wedge F_2 & \text{iff} & I \models F_1 \text{ and } I \models F_2 \\
I \models F_1 \vee F_2 & \text{iff} & I \models F_1 \text{ or } I \models F_2 \\
I \models F_1 \rightarrow F_2 & \text{iff} & I \not\models F_1 \text{ or } I \models F_2 \\
I \models F_1 \leftrightarrow F_2 & \text{iff} & (I \models F_1 \text{ and } I \models F_2) \text{ or } (I \not\models F_1 \text{ and } I \not\models F_2) \\
I \models \forall x.F & \text{iff} & \text{for all } v \in D_I, I \triangleleft \{x \mapsto v\} \models F \\
I \models \exists x.F & \text{iff} & \text{there exists } v \in D_I \text{ such that } I \triangleleft \{x \mapsto v\} \models F
\end{array}
$$

Semantics: Predicates

$$\boxed{I \models p(t_1, \cdots, t_n) \text{ iff } \alpha_I[p(t_1, \cdots, t_n)] = true}$$

- Predicates are evaluated recursively.

$$\alpha_I[p(t_1, \cdots, t_n)] = \alpha_I[p](\alpha_I[t_1], \cdots, \alpha_I[t_n])$$

- During evaluating terms, functions are evaluated recursively as well.

$$\alpha_I[f(t_1, \cdots, t_n)] = \alpha_I[f](\alpha_I[t_1], \cdots, \alpha_I[t_n])$$

Semantics: Quantifiers

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

- $J : I \lhd \{x \mapsto v\}$ denotes the x-variant of I. That is, $I : (D_I, \alpha_I)$ and $J : (D_J, \alpha_J)$ agree on everything except possibly the value of the variable x. Technically,
  - $D_I = D_J$, and
  - $\alpha_I[y] = \alpha_J[y]$ for all constant, free variable, function, and predicate symbols y, except possibly x where $\alpha_J[x] = v$.
- In words, "I is an interpretation of $\forall x.F$ iff all x-variants of I are interpreations of F".

$$\boxed{I \models \exists x.F \text{ iff there exists } v \in D_I \text{ such that } I \lhd \{x \mapsto v\} \models F}$$

- "I is an interpretation of $\exists x.F$ iff some x-variant of I is an interpreation of F".

Example 1: Semantics

Consider the formula

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

and the interpretation $I : (\mathbb{Z}, \alpha_I)$ where

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

The truth value of F under I is computed as follows:

$$\begin{array}{lll} \textbf{1.} & I \models x + y > z & \text{since } \alpha_I[x + y > z] = 13_{\mathbb{Z}} +_{\mathbb{Z}} 42_{\mathbb{Z}} >_{\mathbb{Z}} 1_{\mathbb{Z}} = \mathit{true} \\ \textbf{2.} & I \models y > z - x & \text{since } \alpha_I[y > z - x] = 42_{\mathbb{Z}} +_{\mathbb{Z}} 1_{\mathbb{Z}} >_{\mathbb{Z}} 13_{\mathbb{Z}} = \mathit{true} \\ \textbf{3.} & I \models F & \text{by } \mathbf{1}, \mathbf{2}, \text{ and the semantics of } \to \end{array}$$

Example 2: Semantics

Consider the formula

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

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

$$\alpha_I : \left\{ \begin{array}{lll} f & \mapsto & \{v_1 \mapsto v_1, v_2 \mapsto v_2\}, \\ g & \mapsto & \{v_1 \mapsto v_2, v_2 \mapsto v_1\}, \\ = & \mapsto & \{(a, b) \mapsto \mathit{true} \textbf{ if } a \text{ syntactically equals } b \textbf{ else } \mathit{false}\} \end{array} \right\}$$

Compute the truth value of F under I.

Let J be the x-variant of I, i.e., $J : I \triangleleft \{x \mapsto v\}$ for some $v \in D$.

$$\begin{array}{lll} \textbf{1.} & J \not\models f(x) = g(x) & \text{For any } v \in D,\ \alpha_J[f(x) = g(x)] = \mathit{false} \\ \textbf{2.} & I \not\models \exists x. f(x) = g(x) & \text{by } \mathbf{1} \text{ and the semantics of } \exists \end{array}$$

Summary

- Syntax and semantics of FOL.