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

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

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.
- **free**(F) and $\mathbf{bound}(F)$ denote the free and bounbd variables of F, respectively.
- It is possible that **free**$(F) \cap \mathbf{bound}(F) \neq \emptyset$.
  - Given $F : \forall x.p(f(x), y) \rightarrow \forall y.p(f(x), y)$, **free**$(F) = \{y\}$ and $\mathbf{bound}(F) = \{x, y\}$.
- A formula F is closed if F has no free variables.
- Suppose **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 ($\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$) are handled in the same way as in PL.
- The semantics of predicates and quantifiers are new.

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

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

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

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

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:

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

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 $v \in D$.

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

Satisfiability and Validity

- A formula $\boldsymbol{F}$ is *satisfiable* iff there exists an interpretation $\boldsymbol{I}$ such that $\boldsymbol{I} \models \boldsymbol{F}$.
- A formula $\boldsymbol{F}$ is *valid* iff for all interpretations $\boldsymbol{I}$, $\boldsymbol{I} \models \boldsymbol{F}$.
- Technically, satisfiability and validity are defined for *closed* FOL formulas.
- But we allow two conventions for a formula $\boldsymbol{F}$ with free variables ($\mathbf{free}(\boldsymbol{F}) \neq \emptyset$).
  - If we say that a formula $\boldsymbol{F}$ is valid, we mean that its universal closure $\forall * .\boldsymbol{F}$ is valid.
  - If we say that $\boldsymbol{F}$ is satisfiable, we mean that its existential closure $\exists * .\boldsymbol{F}$ is satisfiable.
- Satisfiability and validity are dual as in PL.

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

Extension of the Semantic Argument Method

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

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

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

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

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

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

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

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:

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

These rules are applied using a domain element v that has *not* been previously used in the proof.

- Why? Given $\exists x.F$, we choose a new value v since we do not know which value in particular satisfies F.

Contradiction Rule

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

- 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.
- A branch is **closed** if it contains a contradiction according to the contradiction rule. It is **open** otherwise.
  - In a finished proof of a valid formula, all branches must be closed.

Example 1: Semantic Argument Method

Determine the validity of the formula F.

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

Suppose F is invalid.

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.	$I \models \bot$	4 and 5

Example 2: Semantic Argument Method

Determine the validity of the formula $\boldsymbol{F}$.

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

Example 2: Semantic Argument Method

Determine the validity of the formula F.

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

We need to show both 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.

$$\begin{array}{lll}
\textbf{1.} & I \models \forall x.p(x) & \text{assumption} \\
\textbf{2.} & I \not\models \neg\exists x.\neg p(x) & \text{assumption} \\
\textbf{3.} & I \models \exists x.\neg p(x) & \text{2 and } \neg \\
\textbf{4.} & I \triangleleft \{x \mapsto v\} \models \neg p(x) & \text{3 and } \exists\text{, for some } v \in D_I \\
\textbf{5.} & I \triangleleft \{x \mapsto v\} \models p(x) & \text{1 and } \forall \\
\textbf{6.} & I \models \bot & \text{4 and 5}
\end{array}$$

Example 2: Semantic Argument Method (cont'd)

Determine the validity of the formula F.

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

We need to show both 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_2 is not valid.

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

Example 3: Semantic Argument Method

Determine the validity of the formula F.

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

Example 3: Semantic Argument Method

Determine the validity of the formula F.

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

Assume F is invalid.

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.	$J : I \triangleleft \{x \mapsto \alpha_I[a]\} \not\models p(x)$	3 and $\exists$
5.	$I \models \bot$	2 and 4

Note that 2 and 4 are contradictory, since $\alpha_I[a] = \alpha_J[x]$.

Example 4: Semantic Argument Method

Show that the formula F is invalid.

$$(\forall x.p(x,x)) \to (\exists x.\forall y.p(x,y))$$

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 proven formula is valid.
- A proof system 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)

Every valid formula F *has a semantic argument proof.*

Substitution

- A **substitution** is a map from FOL formulas to FOL formulas.

$$\sigma : [F_1 \mapsto G_1, \cdots, F_n \mapsto G_n]$$

- To compute $F\sigma$, replace each occurence of F_i in F by G_i simultaneously.
- For example, consider the formula F and the substitution σ

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

Then,

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

Safe Substitution

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

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

Then, safe substitution proceeds as follows.

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

Theorems for Safe Substitution

A FOL version of substitution of equivalent formulas:

Theorem

Consider the substitution

$$\sigma : \{F_1 \to G_1, \cdots, F_n \to G_n\}$$

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

Theorem

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

- formula schema: formula templates with at least one placeholder such as $F_1, F_2, \cdots$.
- side conditions: conditions specifying that certain variables do not occur free in the placeholders.

Examples: Valid Templates

- Consider the valid formula schema

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

  Then, 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 the valid formula schema

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

  Then, 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 (NNF)

- The normal forms of PL extend to FOL.
- A FOL formula F can be transformed into NNF by using the following equivalences.

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

Example: NNF

Convert the formula G into NNF where

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

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

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

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 the equivalence $\neg(F_1 \land F_2) \iff \neg F_1 \lor \neg F_2$.

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

Prenex Normal Form (PNF)

- A formula is in PNF if all of its quantifiers appear at the beginning of the formula:
$$\mathbf{Q}_1 x_1 \cdots \mathbf{Q}_n x_n . F[x_1, \cdots, x_n]$$
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 in front of F_3:
  $$F_4 : Q_1 x_1 \cdots Q_n x_n . F3$$
  where Q_i are the quantifiers such that if Q_j is in the scope of Q_i in F_1, then $i < j$.
- A FOL formula is in CNF (resp., DNF) if (1) it is in PNF and (2) its main quantifier-free subformula is in CNF (resp., DNF).

Example: PNF

Convert the formula F into PNF form.

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

1. Convert into 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 in front of F_3.

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

Decidability

- Satisfiability can be formalized as a decision problem in formal languages.
  - 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) answers "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

FOL is an extension of propositional logic (PL) with predicates, functions, and quantifiers.

- Syntax and semantics of FOL
- Satisfiability and validity
- Substitution, Normal forms