3. Predicate logic

The propositional logic only allows to make statements about specific objects, but it does not allow us to make universal statements and existential statements.

For example, the following are universal statements:

(1) Every student of Hosei University is happy.

(2) Nobody knows what to happen tomorrow.
The following are existential statements:

(1) One of my classmates received an award.
(2) Some students in my class do not like mathematics.
3.1 Predicates

Definition 3.1 A predicate is a truth-valued function.

In other words, a predicate is a function from a domain $X$ to the boolean type $\texttt{bool}$:

$$p: X \rightarrow \texttt{bool}$$
For example:

\[ x > 10 \] is a predicate, but not proposition.
\[ 5 > 10 \] is a proposition, derived from the predicate \( x > 10 \) by substituting 5 for \( x \).

where \( x \) is an integer variable.
Basic types (sets) in SOFL

The following are basic types in SOFL:

- **nat0**: 0, 1, 2, 3, 4, … (natural numbers including 0)
- **nat**: 1, 2, 3, 4, 5, … (natural numbers)
- **int**: … -2, -1, 0, 1, 2, … (integers)
- **real**: … -2.5, -1.4, 0, 1.4, 2.5, (real numbers)
- **char**: ‘a’, ‘b’, ‘x’, ‘%’, … (characters)
- **bool**: true, false (boolean values)
Example: a predicate ``is_big'' is defined as follows:

```python
is_big(x: int): bool
== x > 2000
```

Then, the following propositions can be formed:

- `is_big(2000)` (false)
- `is_big(2004)` (true)
- `is_big(1996)` (false)
3.2 Quantifiers

1. Universal quantifier

For example:

\[ \text{is-big}(x) \equiv x > 10 \] (is-big is a predicate)

Then we can write the conjunction

\[ \text{is-big}(12) \ \text{and} \ \text{is-big}(15) \ \text{and} \ \text{is-big}(20) \]

as

\[ \forall [x: \{12, 15, 20\}] \mid \text{is-big}(x) \]
In general, the universally quantified expression has the form:

\[
\text{forall}[x_1: X_1, \ldots, x_n: X_n] \mid p(x_1, x_2, \ldots, x_n)
\]

forall --- universal quantifier

\(x_i: X_i \ (i = 1..n)\) --- bindings

\(x_1, x_2, \ldots, x_n\) --- bound variables

\(X_1, X_2, \ldots, X_n\) --- the ranges (sets or types) of the bound variables

\(p(x_1, x_2, \ldots, x_n)\) --- predicate
2. Existential quantifier

For example, we can write the disjunction

\[ \text{is-big}(5) \text{ or is-big}(12) \text{ or is-big}(15) \]

as

\[ \text{exists}[x: \{5, 12, 15\}] | \text{is-big}(x) \]

We call such an expression existentially quantified expression.

\[ \text{exists!}[x: T] | p(x) \]

means that there exists an unique \( x \) in \( T \) that satisfies condition \( p(x) \).
In general, the existentially quantified expression has the form:

\[
\text{exists}[x_1: X_1, \ldots, x_n: X_n] \mid p(x_1, x_2, \ldots, x_n)
\]

**exists** --- existential quantifier

\(x_i: X_i\) (i=1..n) --- bindings

\(x_1, x_2, \ldots, x_n\) --- bound variables

\(X_1, X_2, \ldots, X_n\) --- the ranges (sets or types) of the bound variables

\(p(x_1, x_2, \ldots, x_n)\) --- predicate
3. The convention

The body of a quantified expression is considered to extend as far as the right as possible.

Example: the quantified expression

\[
\text{forall}[x: \text{nat}] \mid (x > z \text{ and } (\exists y: \text{nat} \mid y > x))
\]

is equivalent to the expression:

\[
\text{forall}[x: \text{nat}] \mid x > z \text{ and } \exists y: \text{nat} \mid y > x
\]
3.3 Multiple quantifiers

Examples:

forall[x: X] | forall[y: Y] | p(x,y)

can be written as:

forall[x: X, y: Y] | p(x, y)
\(\forall x: X \mid \exists y: Y \mid p(x, y)\)

can be written as:

\(\forall x: X \exists y: Y \mid p(x, y)\)

\(\exists x: X \mid \exists y: Y \mid p(x, y)\)

can be written as:

\(\exists x: X, y: Y \mid p(x, y)\)
Examples of multiple quantifiers

\(\forall i: \text{nat} \; \exists j: \text{nat} \mid j > i\)

This predicate is true, but the inversion of the universal quantifier and the existential quantifier will change the truth of the expression. Consider the following quantified expression, which is not true.

\(\exists j: \text{nat} \; \forall i: \text{nat} \mid j > i\)
3.4 de Morgan’s laws

The quantified expressions satisfy de Morgan's laws. That is,

\[
\text{forall}[x: X] \mid P(x) \\
\Rightarrow [\text{exists-deM}] \\
\text{not (exists}[x: X] \mid \text{not } P(x))
\]

\[
\text{not (forall}[x: X] \mid P(x)) \\
\Rightarrow [\text{forall-deM}] \\
\text{exists}[x: X] \mid \text{not } P(x)
\]
3.5 Substitution

Substitution is an operation that changes a predicate by substituting a variable or expression for a free variable in the predicate.

Let P be a predicate, we use

$$P[x/y]$$

to denote the predicate obtained by substituting variable x for every free occurrence of y in P.
Examples:

\[(x > 5 \text{ and } y > x)[t/x] \iff (t > 5 \text{ and } y > t)\]

\[(10 > 20)[y/x] \iff (10 > 20)\]

\[(x < 20 + y)[(2+z)/y] \iff (x < 20 + (2 + z))\]
Bound variables should not be substituted.

(1) \((\forall x : \text{nat}) \mid x + 1 > 0)[y/x]\iff (\forall x : \text{nat}) \mid x + 1 > 0\)

(2) \((\exists y : \text{nat}) \mid y > x \text{ and } y < x + 15)[5/x]\iff (\exists y : \text{nat}) \mid y > 5 \text{ and } y < 5 + 15\)
Substitutions should not cause confusions.

Example: consider the following substitution:

\[(y > 10 \text{ and } \exists x: \text{nat} \mid x > y)[x/y]\]

where \(y\) is a free variable while \(x\) is a bound variable. The result of the substitution is:

\[x > 10 \text{ and } \exists x: \text{nat} \mid x > x\]

where there is a confusion about variable \(x\): is it a bound variable or free variable? Also, a contradiction \(x > x\) is introduced.
An appropriate way to do the substitution is first to change the name of the bound variable $x$ to another name, say $i$, and then to carry out the substitution.

$$(y > 10 \text{ and } \exists [i: \text{nat}] \mid i > y)[x/y]$$

The result of the substitution becomes:

$$(x > 10 \text{ and } \exists [i: \text{nat}] \mid i > x)$$

In this expression, $i$ is a bound variable while $x$ is a free variable.
Substitution sequence

We use

\[ P[x/y][t/x] \]

to denote the predicate resulting from first substituting \( x \) for occurrences of \( y \) in \( P \) and then substituting \( t \) for occurrences of \( x \) in the predicate \( P[x/y] \).
Example:

\[(y > 10 \text{ and exists}[i: \text{ nat}] \mid i > y)[x/y][t/x] \iff (x > 10 \text{ and exists}[i: \text{ nat}] \mid i > x)[t/x] \iff (t > 10 \text{ and exists}[i: \text{ nat}] \mid i > t)\]
Multiple substitutions

We use

\[ P[x/y, t/z] \]

to denote the predicate resulting from simultaneously substituting \( x \) for occurrences of \( y \) and substituting \( t \) for occurrences of \( z \).

Example:

\[
(y > z + 10 \text{ and } \exists i[: \text{nat}] \mid i > y + z)[x/y, t/z] \\
\leq \Rightarrow \\
(x > t + 10 \text{ and } \exists i[: \text{nat}] \mid i > x + t)
\]
3.6 Proof in predicate logic

(1) Since predicate logic is an extension of propositional logic, all the inference rules of propositional logic are applicable to proof in predicate logic.

(2) There is a need to provide rules for dealing with quantified expressions.
3.6.1 Introduction and elimination of existential quantifiers.

\[ m \text{ inset } X, \ P(x)[m/x] \]
\[ \exists x : X \mid P(x) \]

\[ \exists x : X \mid P(x), \]
\[ m \text{ inset } X \text{ and } P(x)[m/x] \vdash Q \]
\[ Q \]
3.6.2 Introduction and elimination of universal quantifiers

\[ \forall x: X \mid P(x) \]

\[ \forall x: X \mid P(x), \text{ m inset } X \]

\[ \frac{}{\forall x: X \mid P(x), \text{ m inset } X \Rightarrow P(m/x)} \]
3.6.3 The boxed proof in predicate logic

Suppose that we want to prove the truth of the sequent:
\[ y \text{ inset } X, \text{ not } P[y/x] \vdash \text{ not } (\forall x : X | P(x)) \]

We can give a boxed proof for the truth of the sequent:

<table>
<thead>
<tr>
<th>from</th>
<th>y inset X, not P[y/x]</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>exists[x: X]</td>
<td>not P(x)</td>
</tr>
<tr>
<td>infer</td>
<td>not (forall[x: X]</td>
<td>P(x))</td>
</tr>
</tbody>
</table>
3.7 Validity and satisfaction

Definition 3.2 A predicate is valid if it evaluates as true for whatever values of the free variables involved.

For example, let $x$ be a variable over the type int. Then the predicate

$$x > 0 \text{ or } x \leq 0$$

is valid, because it evaluates as true no matter what integer variable $x$ takes.
Definition 3.3 A predicate is **satisfiable** if it evaluates as true for some values of the free variables involved. Otherwise, the predicate is unsatisfiable.

For instance, let $x: \text{int}$, the predicate
\[
x > 10
\]
is satisfiable, since it evaluates as true for some integers, say 15.

On the other hand, the predicate
\[
x > 10 \text{ and } x < 10
\]
is unsatisfiable, since it evaluates as false for whatever integers bound to $x$. 
The correspondence between notions in both propositional logic and predicate logic:

A predicate is valid \( \iff \) tautology
A predicate is satisfiable \( \iff \) contingency
A predicate is unsatisfiable \( \iff \) contradiction
Definition 3.4 If a predicate may not yield a truth value for some values bound to its free variables, we call the predicate partial predicate.

For example,

\[ x / 0 > 5 \]

is a partial predicate.

The problem is that predicate logic does not allow undefinedness to join evaluation of predicates.
One way to deal with this problem is to extend the truth tables of all the logical operators (i.e., and, or, not, =>, <=>) to allow the special value “undefinedness” to participate in evaluations of predicates.

The extension is made in the way that a result is given whenever possible, according to the predicate logic.
<table>
<thead>
<tr>
<th>(and)</th>
<th>true</th>
<th>nil</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>nil</td>
<td>false</td>
</tr>
<tr>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(or)</th>
<th>true</th>
<th>nil</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>nil</td>
<td>true</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>nil</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(not)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Examples:

nil and false <=> false
nil and true <=> nil
nil and nil  <=> nil
<table>
<thead>
<tr>
<th>$(=&gt;)$</th>
<th>true</th>
<th>nil</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>nil</td>
<td>false</td>
</tr>
<tr>
<td>nil</td>
<td>true</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$&lt;=$</th>
<th>true</th>
<th>nil</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>nil</td>
<td>false</td>
</tr>
<tr>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>nil</td>
<td>true</td>
</tr>
</tbody>
</table>
3.9 Formal specification with predicates

Predicate expressions are usually used to describe conditions or properties, but if they are used in a specific way, they can help express functional requirements of systems, depending on how they are interpreted.

For example, consider the expression:

\[ x > 5 \text{ and } x < 10 \]

It can be interpreted as a condition requiring the input \( x \) to be less than 10 and greater than 5, but if \( x \) is an output of the system, the expression may also be interpreted as a condition requiring the system to produce such an \( x \) that is less than 10 and greater than 5.
Exercise 3

1. Answer the following questions:
   a. what is the similarity and difference between a predicate and a function?
   b. what is the difference between a universally quantified expression and existentially quantified expression?
   c. what is a substitution?
   d. what is a valid predicate?
   e. what is a satisfiable predicate?
   f. what is a partial predicate?
2. Tell which of the following quantified predicate expressions are propositions.

a. \( \text{forall} [x: \text{int}] \mid x > 5 \text{ and } x < 10 \)

b. \( \text{exists} [x: \text{int}] \mid y > x \text{ and } y < x + 10 \)

c. \( \text{forall} [x, y: \text{real}] \mid x + y > x - y \)

d. \( \text{forall} [x, y: \text{real}] \text{exists} [z: \text{real}] \mid x + y > z \)

e. \( \text{exists} [x: \text{int}] \text{forall} [y: \text{int}] \mid x \ast y > z \)
3. Evaluate the substitutions.

a. \((x > y + z \Rightarrow y < x)[t/x]\)

b. \((\forall[x, y: \text{nat0}] \mid x < z \land z < y \Rightarrow x < y)[m/y, t/z]\)

c. \((\exists[x, y, z: \text{nat}] \mid x \times y > z \Rightarrow x > z \land y > z \land b > c)[a/x, b/y, c/b]\)
4. Tell which predicates are true according to the extended truth tables.

a. $x > y$ and $y / 0 > 5 \iff$ false
b. $x > y$ and $y > x \iff$ nil
c. true or nil $\iff$ nil
d. false or nil $\iff$ false
e. false $\Rightarrow$ nil $\iff$ nil
f. true $\Rightarrow$ false $\iff$ nil
g. true $\Rightarrow$ nil $\iff$ false
h. true $\iff$ false $\iff$ nil
i. false $\iff$ nil $\iff$ true