Chapter 1e
Predicate Logic and Program Verification

Discrete Mathematics II

(Materials drawn from **Chapter 2** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006.")

Nguyen An Khuong, Huynh Tuong Nguyen
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

Warm-up questions

a) Are there expressions in Predicate Logic that do not evaluate to TRUE or FALSE? If so, give an example.

Ans.: Terms, unlike predicates and formulas, do not evaluate to the distinguished symbols true or false. Examples of terms include: a, a constant (or 0-ary function); x, a variable; $f(t)$, a unary function f applied to a term t.

b) Is $p(a) \longrightarrow \exists x.p(x)$ a valid formula?

Ans.: Yes

c) How do you represent a propositional variable (as used in Propositional Logic) in a Predicate Logic formula?

Ans.: As a 0-ary predicate.

d) Fermat's Last Theorem is the name of the statement in number theory that: *It is impossible to separate any power higher than the second into two like powers.*

Or, more precisely:

If an integer n is greater than 2, then the equation $x^n + y^n = z^n$ has no solutions in positive integers x, y, and z.

Formulate the above statement in Predicate Logic with Equality?

Warm-up questions (cont'd): An answer to Fermat's Last Theorem Formulation

$$\forall n.integer(n) \land n > 2 \longrightarrow \forall x, y, z.integer(x) \land integer(y) \land integer(z) \land x > 0 \land y > 0 \land z > 0 \longrightarrow x^n + y^n \neq z^n.$$

Program Verification

- Below is a function written in an imperative programming language to perform *binary search*, by returning TRUE iff the array a contains the value e in the range $[l, u]$ and FALSE otherwise, under the assumption that the input range is sorted.

```
bool binarySearch ( int [] a, int l, int u, int e) {
if (l > u) return false ;
else {
int m = (l + u) div 2;
if (a[m] == e) return true ;
else if (a[m] < e) return binarySearch (a, m + 1, u, e);
else return binarySearch (a, l, m - 1, e);
}
}
```

- As a first step towards determining whether an implementation (such as that in the function above) fulfills its specification, the specification has to be formalized. We do so in terms of *preconditions* and *postconditions.*

Program Verification (cont'd)

- A *precondition* specifies what should be true upon entering the function (i.e., under what inputs the function is expected to work).
- The *postcondition* is a formula G whose free variables include only the formal parameters and the special variable rv representing the return value of the function.
- The postcondition relates the function's output (the return value rv) to its input (the parameters).

Prob: Formulate in Predicate Logic the precondition and the postcondition for `binarySearch`.

Program Verification (cont'd): Answer

- First precondition: $0 \leq l \land u < |a|$
- Second precondition:
  $\forall i, j . integer(i) \land integer(j) \land 0 \leq i \leq j < |a| \longrightarrow a[i] \leq a[j]$
- Postcondition: $rv \longleftrightarrow \exists i . l \leq i \leq u \land a[i] = e$

HW

1. Do all HWs which have not been done in previous lectures.
2. Try to understand deeply the following notations/terms

   arity, expression, term, formula, atomic formula, sentence, clause, Backus Naur form (BNF), parse tree, precondition, postcondition, binding priorities, provability, witness, scope, bound, verification, model checking, Hoare triple, and their other related notation/terms.
3. Do exercise 1.5.14 on page 89 in [2].
4. Consider the following program

   ```
   temp := x
   x := y
   y := temp
   ```

   What does this tinny program do? Find preconditions, postconditions and verify its correctness?