Program Verification

In programming, we are becoming more concerned about correctness and reliability in our finished products.

Correctness issues:

1. Given program P, what does it mean ? I.e., what is its **specification** S ?

2. Given specification S, design program P that implements it.

3. Do specification S and program P perform the same function ?

History: 3 in 1970s, 1 in 1980s, and today 2.

- Testing verifies correctness 100% only for the test instances.

- A program computes a function.

- Suppose a specification S of the function is given as part of the initial design of the program.

- Suppose that it is possible to determine what function a given program P computes.

- If it were possible to show that the function computed by the program is in fact the same function given by the specification, then we have correctness of the program.

- **Predicate Calculus** (First Order Logic) is particularly useful for specifying complex functions precisely.

- Notation: $\{\alpha\}P\{\beta\}$ means that if formula α is true before execution of P, P is executed and terminates, then formula β will be true.

- **Rules:**

`consequence right:` $$\frac{\{\alpha\}P\{\beta\}, (\beta \supset \gamma)}{\{\alpha\}P\{\gamma\}}$$

`consequence left:` $$\frac{(\gamma \supset \alpha), \{\alpha\}P\{\beta\}}{\{\gamma\}P\{\beta\}}$$

`composition:` $$\frac{\{\alpha\}P_1\{\beta\}, \{\beta\}P_2\{\gamma\}}{\{\alpha\}P_1P_2\{\gamma\}}$$

$$\texttt{assignment:}\quad \frac{x = t}{\{\alpha(t)\}x = t\{\alpha(x)\}}$$

$$\texttt{if:}\quad \frac{\{\alpha \wedge \beta\}P_1\{\gamma\}, \{\alpha \wedge \neg\beta\}P_2\{\gamma\}}{\{\alpha\}\ \textbf{if}\ \beta\ \textbf{then}\ P_1\ \textbf{else}\ P_2\ \{\gamma\}}$$

$$\texttt{while:}\quad \frac{\{\alpha \wedge \beta\}P\{\alpha\}}{\{\alpha\}\ \textbf{while}\ \beta\ \textbf{do}\ P\ \{\alpha \wedge \neg\beta\}}$$

If the antecedent in each inference rule can be shown to be true, then we can replace the antecedent with the consequent.

Using these rules, and others, we can show:

$$\{\text{precondition}\}P\{\text{postcondition}\}$$

for a given program P, and thus prove the correctness of P.

Example: program computes $y = A * B$:

$\{B \geq 0\}$

```
mult(int A, int B) {
  int a = A;
  int b = B;
  int y = 0;
  while (b > 0) {
    y = y + a;
    b = b - 1; }}
```

$\{y = A * B\}$

Each pass through the **while** loop adds a to y, but $a * b$ decreases by a because b is decremented by 1. Therefore, we get the loop invariant:

$$(y + (a * b) = A * B) \wedge b \geq 0$$