Necessity of Systematic & Automated Testing Techniques

Moonzoo Kim
CS Dept, KAIST

Remarks by Bill Gates

17th Annual ACM Conference on Object-Oriented Programming, Systems Languages, and Applications, Seattle, Washington, November 8, 2002

- **"… When you look at a big commercial software company like Microsoft, there's actually as much testing that goes in as development. We have as many testers as we have developers. Testers basically test all the time, and developers basically are involved in the testing process about half the time…"**
- **"… We've probably changed the industry we're in. We're not in the software industry; we're in the testing industry, and writing the software is the thing that keeps us busy doing all that testing."**
- **"…The test cases are unbelievably expensive; in fact, there's more lines of code in the test harness than there is in the program itself. Often that's a ratio of about three to one."**

Ex. Testing a Triangle Decision Program

Input : Read three integer values from the command line. The three values represent the length of the sides of a triangle.

Output : Tell whether the triangle is

- 부등변삼각형 (Scalene) : no two sides are equal
- 이등변삼각형(Isosceles) : exactly two sides are equal
- 정삼각형 (Equilateral) : all sides are equal

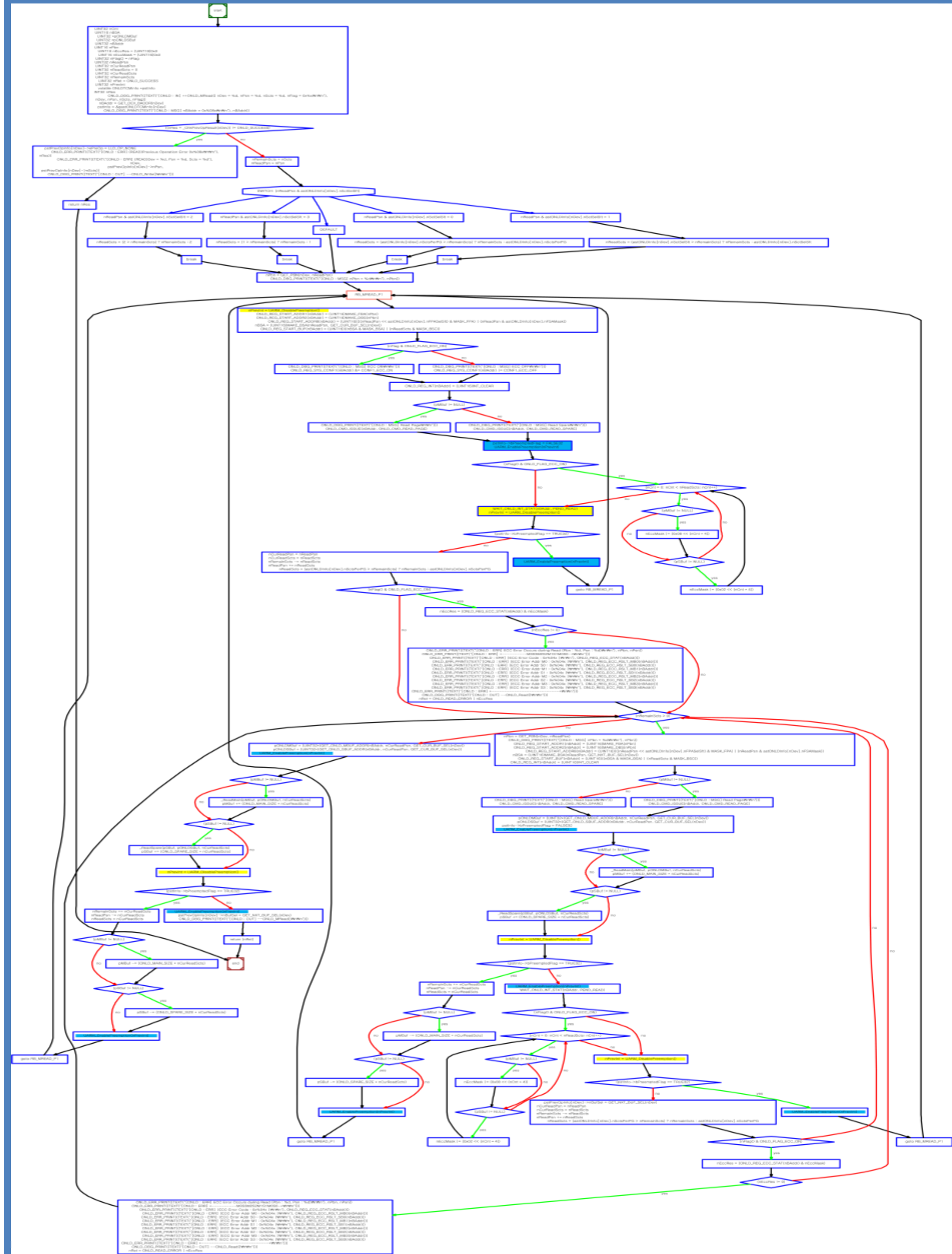
Create a Set of Test Cases for this program

(3,4,5), (2,2,1), (1,1,1) ?

Precondition (Input Validity) Check

- Condition 1: $a > 0, b > 0, c > 0$
- Condition 2: $a < b + c$
  - Ex. (4, 2, 1) is an invalid triangle
  - Permutation of the above condition
    - $a < b + c$
    - $b < a + c$
    - $c < a + b$
- What if $b + c$ exceeds 2^{32} (i.e. overflow)?
  - long v.s. int v.s. short. v.s. char
- Developers often fail to consider implicit preconditions
  - Cause of many hard-to-find bugs

- # of test cases required?
  ① 4
  ② 10
  ③ 50
  ④ 100
- # of feasible unique execution paths?
  - 11 paths
  - guess what test cases needed



More Complex Testing Situations (1/3)

- Software is constantly changing
  - What if "integer value" is relaxed to "floating value" ?
    - Round-off errors should be handled explicitly
  - What if new statements S_1 … S_n are added to check whether the given triangle is 직각삼각형 (a right angle triangle)?
    - Will you test all previous tests again?
    - How to create minimal test cases to check the changed parts of the target program

More Complex Testing Situations (2/3)

- Regression testing is essential
  - How to select statements/conditions affected by the revision of the program?
  - How to create test cases to cover those statements/conditions?
  - How to create efficient test cases?
    - How to create a minimal set of test cases (i.e. # of test cases is small)?
    - How to create a minimal test case (i.e. causing minimal execution time)?
  - How to reuse pre-existing test cases?

More Complex Testing Situations (3/3)

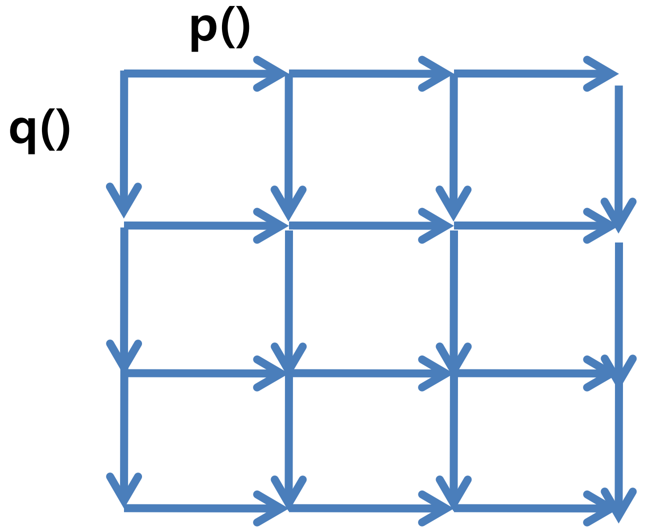
- However, conventional coverage is not complete
  - Ex. Int adder(int x, int y) { return 3;}
    - Test case (x=1,y=2) covers all statements/branches of the target program and detects no error
    - In other words, all variable values must be explored for complete results
- Formal verification aims to guarantee completeness
  - Model checking analyzes all possible x, y values through 2^{64} (=2^{32} x 2^{32}) cases
  - However, model checking is more popular for debugging, not verification

Concurrency

- Concurrent programs have very high complexity due to non-deterministic scheduling
  - Ex. int x=0, y=0, z =0;

    void p() {x=y+1; y=z+1; z= x+1;}

    void q() {y=z+1; z=x+1; x=y+1;}



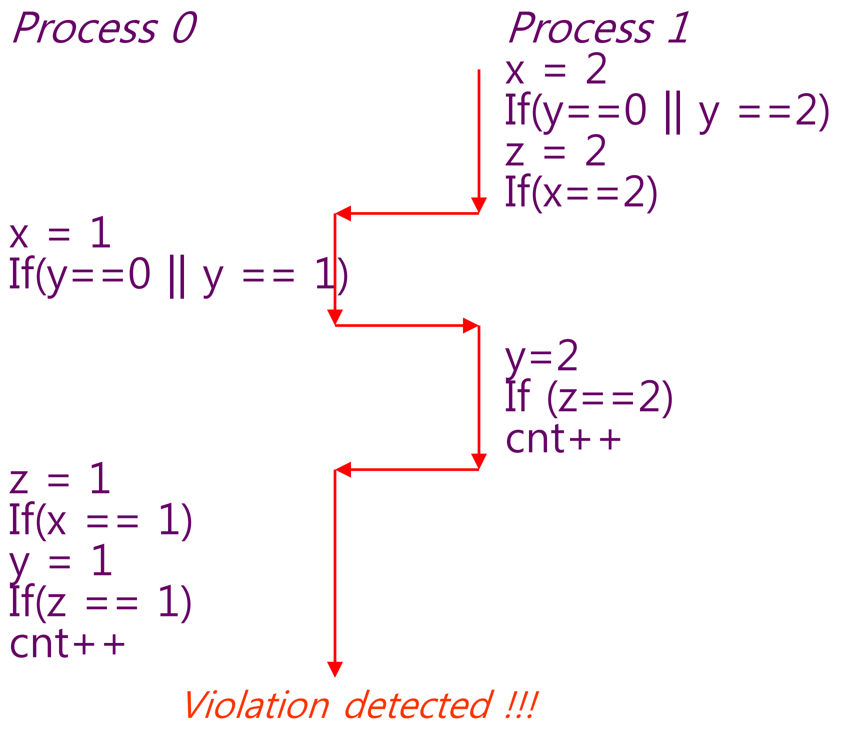
An Example of Mutual Exclusion Protocol

```
char cnt=0,x=0,y=0,z=0;

void process() {
        char me=_pid +1; /* me is 1 or 2*/
again:
        x = me;
        If (y ==0 || y== me) ;
        else goto again;

        z =me;
        If (x == me) ;
        else goto again;

        y=me;
        If(z==me);
        else goto again;

        /* enter critical section */
        cnt++;
        assert( cnt ==1);
        cnt --;
        goto again;
}
```

Software locks

Critical section

Mutual Exclusion Algorithm



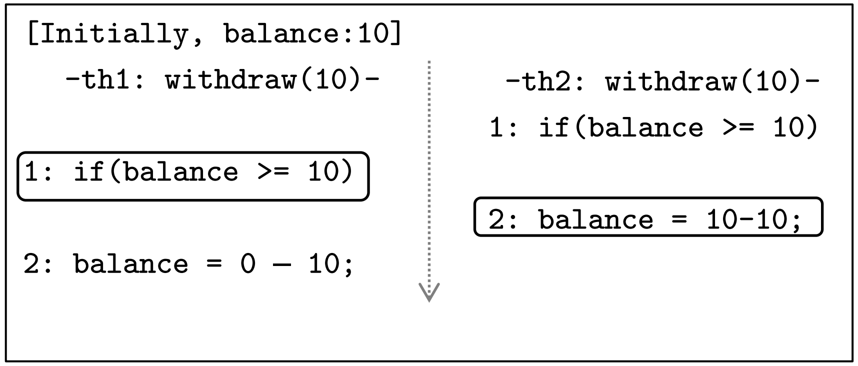
Counter Example

More Concurrency Bugs

- Data race bugs

```
class Account_DR {
  double balance;
  // INV:balance should be always non-negative

   void withdraw(double x) {
1:   if (balance >= x) {
2:     balance = balance–x;}
       ...
   }}
```

(a) Buggy program code



```
[Initially, balance:10]
   -th1: withdraw(10)-            -th2: withdraw(10)-
                                 1: if(balance >= 10)
1: if(balance >= 10)
                                 2: balance = 10-10;
2: balance = 0 – 10;
```

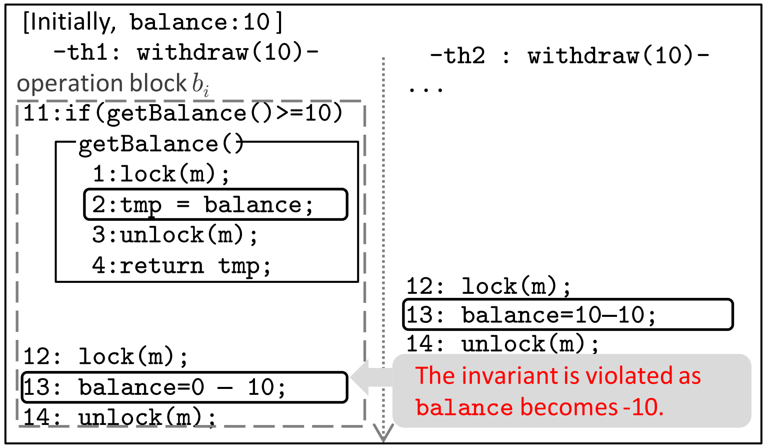
The invariant is violated as balance becomes -10.

(b) Erroneous execution

- Atomicity bugs

```
class Account_BR {
  Lock m;
  double balance;
  // INV: balance should be non-negative

  double getBalance() {   void withdraw(double x){
    double tmp;             /*@atomic region begins*/
1:  lock(m);              11: if (getBalance() >= x){
2:  tmp = balance ;       12:   lock(m);
3:  unlock(m);            13:   balance = balance – x;
4:  return tmp; }         14:   unlock(m); }
                            /*@atomic region ends*/
                            ... }
```

(a) Buggy program code



```
[Initially, balance:10]
  -th1: withdraw(10)-            -th2 : withdraw(10)-
operation block b_i              ...
11:if(getBalance()>=10)
    getBalance()
      1:lock(m);
      2:tmp = balance;
      3:unlock(m);
      4:return tmp;
                                 12: lock(m);
                                 13: balance=10–10;
                                 14: unlock(m);
12: lock(m);
13: balance=0 – 10;
14: unlock(m);
```

The invariant is violated as balance becomes -10.

(b) Erroneous execution

Summary

1. Software = a large set of unique executions
2. SW testing = to find an execution that violates a given requirement among the large set
   – A human brain is poor at enumerating all executions of a target SW, but computer is good at the task
3. Automated SW testing
   = to enumerate and analyze the executions of SW systematically (and exhaustively if possible)