Lecture 3
Code Coverage (I)
Last class

- Software Testing
  - Concepts
  - Granularity
  - Unit Testing
- JUnit
This class

• Code coverage
  • Control-flow coverage
    • Statement coverage
    • Branch coverage
    • Path coverage
  • Coverage Collection Tools
    • EclEmma
Who will test the tests?

- Code coverage can be a way!
  - Usually, a test covering/executing more code may indicate better test quality
How to measure code coverage
Overview

- A common way is to abstract program into graphs
  - Graph: Usually the control flow graph (CFG)
  - Node coverage: Execute every statement
  - Edge coverage: Execute every branch
Control Flow Graphs

- A CFG models all executions of a program by describing control structures
  - Node: Sequences of statements (basic block)
  - Basic Block: A sequence of statements with only one entry point and only one exit point (no branches)
  - Edge: Transfers of control
if \((x < y)\) 
{} 
\(y = 0;\) 
\(x = x + 1;\) 
}
else 
{} 
\(x = y;\) 
}
...

the two statements can be in the same nodes because there is no branch between them.
if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{
    x = y;
}
...

if (x < y)
{
    y = 0;
    x = x + 1;
}
else
{...
if \((x < y)\) {
    return;
}
print \((x)\);
return;

CFG : The Dummy Nodes

Some program may have multiple exit nodes!
CFG : while and for Loops

```plaintext
x=0;
while (x < y) {
  y = f (x, y);
  x = x + 1;
}
...

for (x = 0; x < y; x++) {
  y = f (x, y);
}
```
x = 0;
while (x < y)
{
    y = f(x, y);
    x = x + 1;
}
...

for (x = 0; x < y; x++)
{
    y = f(x, y);
}

x = 0;
do {
    y = f(x, y);
    x = x + 1;
} while (x < y)
...

CFG : while and for Loops
while (x < y)
{
    y = f (x, y);
    if (y == 0) {
        break;
    } else if (y<0) {
        y = y*2;
        continue;
    }
    x = x + 1;
}
print (y);
read ( c ) ;
switch ( c )
{
    case ‘N’:
        y = 25;
        break;
    case ‘Y’:
        y = 50;
        break;
    default:
        y = 0;
        break;
}
print (y);
public class CFGCoverageExample {
    public int testMe(int x, boolean a, boolean b) {
        if (a)
            x++;  
        if (b)
            x--;  
        return x;
    }
}
public class JUnitStatementCov {
    CFGCoverageExample tester;
    @Before
    public void initialize() {
        tester = new CFGCoverageExample();
    }
    @Test
    public void testCase() {
        assertEquals(0, tester.testMe(0, true, false));
    }
}
CFG-based Coverage: Statement Coverage

- The percentage of statements covered by the test

```java
tester.testMe(0, true, false)
```

```java
if(a)
    a==true
    a==false
if(b)
    x++
    b==true
    x--
    b==false
return x
```

```
x=0 a=true b=false
```
CFG-based Coverage: Statement Coverage

- The percentage of statements covered by the test

```
tester.testMe(0, true, false)
```

```
x=0 a=true b=false
```

**SCov=4/5=80%**
CFG-based Coverage: Branch Coverage

- The percentage of branches covered by the test
- Consider both false and true branch for each conditional statement
CFG-based Coverage: Branch Coverage

- The percentage of branches covered by the test
- Consider both false and true branch for each conditional statement

```
if(a)
  a==true
  x++
  a==false
if(b)
  x--
  b==true
  b==false
return x
```

BCov=2/4=50%
CFG-based Coverage: Path Coverage

- The percentage of paths covered by the test
- Consider all possible program execution paths

```cpp
if(a)
  if(b)
    x++
  else
    x--
else
  return x
```
CFG-based Coverage: Path Coverage

- The percentage of paths covered by the test
- Consider all possible program execution paths
CFG-based Coverage: Path Coverage

- The percentage of paths covered by the test
- Consider all possible program execution paths
CFG-based Coverage: Path Coverage

• The percentage of paths covered by the test
  • Consider all possible program execution paths
CFG-based Coverage: Path Coverage

- The percentage of paths covered by the test
- Consider all possible program execution paths

PCov = 1/4 = 25%
public class JUnitStatementCov {
    CFGCoverageExample tester;
    @Before
    public void initialize() {
        tester = new CFGCoverageExample();
    }
    @Test
    public void testCase() {
        assertEquals(0, tester.testMe(0, true, false));
    }
}

Statement coverage: 80%
Branch coverage: 50%
Path coverage: 25%

If we achieve 100% branch coverage, do we get 100% statement coverage for free?

If we achieve 100% path coverage, do we get 100% branch coverage for free?
Statement Coverage VS. Branch Coverage

• If a test suite achieve 100% b-coverage, it must achieve 100% s-coverage
  • The statements not in branches will be covered by any test
  • All other statements are in certain branch
• If a test suite achieve 100% s-coverage, will it achieve 100% b-coverage?
Statement Coverage VS. Branch Coverage

- If a test suite achieve 100% b-coverage, it must achieve 100% s-coverage
- The statements not in branches will be covered by any test
- All other statements are in certain branch
- If a test suite achieve 100% s-coverage, will it achieve 100% b-coverage?

```
x=0 a=true b=true
```

```
if(a)
  a==true
  a==false

x++
```

```
if(b)
  b==true
  b==false

x--
```

```
return x
```
Statement Coverage VS. Branch Coverage

- If a test suite achieve 100% b-coverage, it must achieve 100% s-coverage.
- The statements not in branches will be covered by any test.
- All other statements are in certain branch.
- If a test suite achieve 100% s-coverage, will it achieve 100% b-coverage?

Branch coverage strictly subsumes statement coverage.
Branch Coverage VS. Path Coverage

• If a test suite achieve 100% p-coverage, it must achieve 100% b-coverage
  • All the branch combinations have been covered indicate all branches are covered
• If a test suite achieve 100% b-coverage, will it achieve 100% p-coverage?
Branch Coverage VS. Path Coverage

- If a test suite achieve 100% p-coverage, it must achieve 100% b-coverage.
- All the branch combinations have been covered indicate all branches are covered.
- If a test suite achieve 100% b-coverage, will it achieve 100% p-coverage?
Branch Coverage VS. Path Coverage

- If a test suite achieve 100% p-coverage, it must achieve 100% b-coverage
- All the branch combinations have been covered indicate all branches are covered
- If a test suite achieve 100% b-coverage, will it achieve 100% p-coverage?

Path coverage strictly subsumes branch coverage
CFG-based Coverage: Comparison Summary

Path coverage strictly subsumes branch coverage strictly subsumes statement coverage

Path Coverage
\[\downarrow\]
Branch Coverage
\[\downarrow\]
Statement Coverage
Should we just use path coverage?

```java
while (x < y)
{
    y = f(x, y);
    x = x + 1;
}
...
```

Possible Paths

- e₁ 
- e₁e₂e₁e₂e₁e₂e₃ 
- e₁e₂e₁e₂e₁e₂e₃ 
- ... 

Path coverage can be infeasible for real-world programs
CFG-based Coverage: Effectiveness

- About 65% of all bugs can be caught in unit testing
- Unit testing is dominated by control-flow testing methods
- Statement and branch testing dominates control-flow testing
CFG-based Coverage: Limitation

• 100% coverage of some aspect is never a guarantee of bug-free software

```java
public int sum(int x, int y){
    return x-y; //should be x+y
}
```

Test: `assertEquals(1, sum(1,0))`

Statement coverage: 100%
Branch coverage: 100%
Path coverage: 100%

Failed to detect the bug...
This class

• Code coverage
  • Control-flow coverage
    • Statement coverage
    • Branch coverage
    • Path coverage
  • Coverage Collection Tools
    • EclEmma
Coverage Collection: Mechanism

- The source code is instrumented (source/binary)
  - Log code that writes to a trace file is inserted in every branch, statement etc.
- When the instrumented code is executed, the coverage info will be written to trace file

```
if(b)
  x--
return x
```

Coverage file:
- line1
- line2
- line3
- line4
Coverage Collection: Tool Supports

- Emma: http://emma.sourceforge.net/
- EclEmma: http://www.eclelemma.org/installation.html/
- Cobertura: http://cobertura.github.io/cobertura/
- Clover: https://www.atlassian.com/software/clover/overview
- JCode: https://wiki.openjdk.java.net/display/CodeTools/jcov
- JaCoCo: http://www.eclelemma.org/jacoco/
EclEmma: Installation

- From your Eclipse menu select Help → Install New Software...
- In the Install dialog enter http://update.eclemma.org/ at the Work with field
- Check the latest EclEmma version and press Next
- Follow the steps in the installation wizard.
EclEmma: Installation

• From your Eclipse menu select Help → Install New Software...

• In the Install dialog enter http://update.eclemma.org/ at the Work with field

• Check the latest EclEmma version and press Next

• Follow the steps in the installation wizard.
EclEmma: Execution

• The installation was successful if you can see the coverage launcher in the toolbar of the Java perspective:

• Coverage collection
  • Right click the test suite class file to run
  • Click “Coverage As” => “JUnit Tests”
EclEmma: Results
Next class

• More on code coverage
Thanks!