Path Testing + Coverage

Chapter 8

Structural Testing

- Also known as glass/white/open box testing
- A software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data
- Functional Testing uses program specification
- Structural Testing is based on specific knowledge of the source code to define the test cases and to examine outputs.

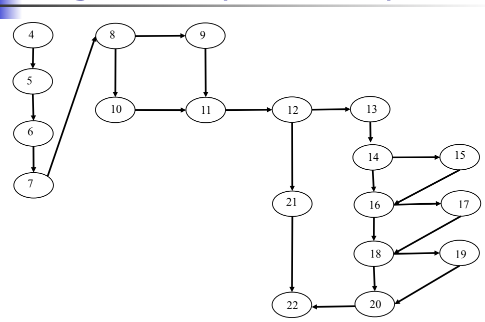
Structural Testing

- Structural testing methods are very amenable to:
  - Rigorous definitions
    - Control flow, data flow, coverage criteria
  - Mathematical analysis
    - Graphs, path analysis
  - Precise measurement
    - Metrics, coverage analysis

Program Graph - Definition

- Given a program written in an imperative programming language, its **program graph** is a directed graph in which nodes are statement fragments, and edges represent flow of control
- A complete statement is also considered a statement fragment

Program Graph - Example



DD-Path

- A decision-to-decision path (DD-Path) is a chain in a program graph such that:
  - Case1: it consists of a single node with indeg=0
  - Case2: it consists of a single node with outdeg=0
  - Case3: it consists of a single node with indeg ≥ 2 or outdeg ≥ 2
  - Case4: it consists of a single node with indeg =1, and outdeg = 1
  - Case5: it is a maximal chain of length ≥ 1
- DD-Paths are also known as **segments**

DD-Path Graph

- Given a program written in an imperative language, its **DD-Path graph** is a directed graph, in which nodes are DD-Paths of its program graph, and edges represent control flow between successor DD-Paths.
- Also known as **Control Flow Graph**

7

Control Flow Graph Derivation

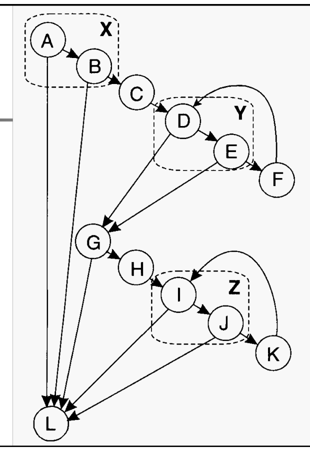
- Straightforward process
- Some judgement is required
- The last statement in a segment must be a predicate, a loop control, a break, or a method exit
- Let’s try an example…

8

```
public int displayLastMsg(int nToPrint) {
  np = 0;
  if ((msgCounter > 0) && (nToPrint > 0)) {
    for (int j = lastMsg; (( j != 0) && (np < nToPrint)); --j) {
      System.out.println(messageBuffer[j]);
      ++np;
    }
    if (np < nToPrint) {
      for (int j = SIZE; ((j != 0) && (np < nToPrint)); --j) {
        System.out.println(messageBuffer[j]);
        ++np;
      }
    }
  }
  return np;
}
```

9

Control flow graph for previous slide



10

Control flow graphs

- Depict which program segments may be followed by others
- A segment is a node in the CFG
- A conditional transfer of control is a **branch** represented by an edge
- An **entry node** (no inbound edges) represents the entry point to a method
- An **exit node** (no outbound edges) represents an exit point of a method

11

Control flow graphs

- An **entry-exit path** is a path from the entry node to the exit node
- **Path expressions** represent paths as sequences of nodes
- Loops are represented as segments within parentheses followed by an asterisk
- There are 22 different path expressions in our example

12

Example path expressions

AL
ABL
ABCDGL
ABCDEGL
ABC(DEF)*DGL
ABC(DEF)*DEGL
ABCDGHIL
ABCDGHIJL
ABCDGH(IJK)*IL
ABC(DEF)*DEGH(IJK)*IJL

13

Code coverage models

- Statement Coverage
- Segment Coverage
- Branch Coverage
- Multiple-Condition Coverage

14

Statement coverage

- Achieved when all statements in a method have been executed at least once
- A test case that will follow the path expression below will achieve statement coverage in our example

  ABC(DEF)*DGH(IJK)*IL

- One test case is enough to achieve statement coverage!

15

Segment coverage

- **Segment coverage** counts segments rather than statements
- May produce drastically different numbers
  - Assume two segments P and Q
  - P has one statement, Q has nine
  - Exercising only one of the segments will give 10% or 90% statement coverage
  - Segment coverage will be 50% in both cases

16

Statement coverage problems

- Predicate may be tested for only one value (misses many bugs)
- Loop bodies may only be iterated once
- Statement coverage can be achieved without branch coverage. Important cases may be missed

```
String s = null;
if (x != y) s = "Hi";
String s2 = s.substring(1);
```

17

Branch coverage

- Achieved when every path from a node is executed at least once
- At least one true and one false evaluation for each predicate
- Can be achieved with D+1 paths in a control flow graph with D 2-way branching nodes and no loops
  - Even less if there are loops

18

Branch coverage problems

- Short-circuit evaluation means that many predicates might not be evaluated
- A compound predicate is treated as a single statement. If n clauses, 2^n combinations, but only 2 are tested
- Only a subset of all entry-exit paths is tested

```
if (a == b) x++;
if (c == d) x--;
```


Multiple-condition coverage

- All true-false combinations of simple conditions in compound predicates are considered at least once
- A truth table may be necessary
- Not necessarily achievable due to lazy evaluation or mutually exclusive conditions

```
if ((x > 0) && (x < 5)) …
```


Dealing with Loops

- Loops are highly fault-prone, so they need to be tested carefully
- Simple view: Every loop involves a decision to traverse the loop or not
- A bit better: Boundary value analysis on the index variable
- Nested loops have to be tested separately starting with the innermost

Creating test cases

- In order to increase the coverage of a test suite, one needs to generate test cases that exercise certain statements or follow a specific path
- This is not always easy to do…

CFG question

- **What is the control flow graph for the following?**

if a < b then c = a + b ; d = a * b
else c = a * b ; d = a + b
if c < d then x = a + c ; y = b + d
else x = a * c ; y = b * d

Creating a test case

- **What is the key question that needs to be answered to be able to create a test for a path?**

Creating a test case

- **What is the key question that needs to be answered to be able to create a test for a path?**
  - How to make the path execute, if possible.
    - Generate input data that satisfies all the conditions on the path.

Creating a test case

- **What are the key items you need to generate a test case for a path?**

Creating a test case

- **What are the key items you need to generate a test case for a path?**
  - Input vector
  - Predicate
  - Path predicate
  - Predicate interpretation
  - Path predicate expression
  - Create test input from path predicate expression

Input Vector

- **What is an input vector?**

Input Vector – 2

- **What is an input vector?**
  - A collection of all data entities read by the routine whose values must be fixed prior to entering the routine.

Input Vector – 3

- **What are the members of an input vector?**

Input Vector – 4

- **What are the members of an input vector?**
  - Input arguments to the routine
  - Global variables and constants
  - Files
  - Network connections
  - Timers

31

Predicate

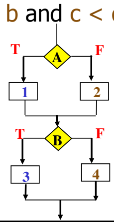
- **What is a predicate?**

32

Predicate – 2

- **What is a predicate?**
- A logical function evaluated at a decision point.
  - In the following each of a < b and c < d are predicates

if a < b **then** c = a + b ; d = a * b
else c = a * b ; d = a + b
if c < d **then** x = a + c ; y = b + d
else x = a * c ; y = b * d



33

Path predicate

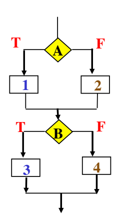
- **What is a path predicate?**

34

Path predicate – 2

- The set of predicates associated with a path.
  - a < b = true ∧ c < d = false is a path predicate

if a < b **then** c = a + b ; d = a * b
else c = a * b ; d = a + b
if c < d **then** x = a + c ; y = b + d
else x = a * c ; y = b * d



35

Path Predicate Expression

- **What is a path predicate expression?**

36

Path Predicate Expression – 2

- **What is a path predicate expression?**
  - An interpreted path predicate

37

Predicate Interpretation

- **What is a path predicate interpretation?**

38

Predicate Interpretation – 2

- **What is a path predicate interpretation?**
  - A path predicate may contain local variables.
  - Local variables cannot be selected independently of the input variables
  - Local variables are eliminated with symbolic execution

39

Predicate Interpretation – 3

- **What is symbolic execution?**
  - Symbolically substituting operations along a path in order to express the predicate solely in terms of the input vector and a constant vector.
  - A predicate may have different interpretations depending on how control reaches the predicate.

40

Attributes of a Path Predicate Expression

- **What are the attributes of a path predicate expression?**

41

Attributes of a Path Predicate Expression – 2

- **What are the attributes of a path predicate expression?**
  - No local variables
  - A set of constraints in terms of the input vector, and, maybe, constants
  - Path forcing inputs are generated by solving the constraints
  - If a path predicate expression has no solution, the path is infeasible

42

Path Predicate Generating Input Values

if a < b **then** c = a + b ; d = a * b
else c = a * b ; d = a + b
if c < d **then** x = a + c ; y = b + d
else x = a * c ; y = b * d

- Path predicate $a < b = true \wedge c < d = false$
- Substitute for c and d $c = a + b \quad d = a * b$

$a < b = true \;\wedge\; a + b < a * b = false$

$\rightarrow \quad a < b \;\wedge\; a + b \geq a * b$

Path Predicate Generating Input Values – 2

$a < b \wedge a + b \geq a * b$

- Solve for a and b $a = 0 \;\wedge\; b = 1$
  - Solutions are not unique
- A solution exists
  - We have a feasible path
- No solution to the constraints
  - Have an infeasible path

Organizing path predicates

- **How can we organize the set of path predicates?**

Organizing path predicates – 2

- **How can we organize the set of path predicates?**
  - Use a decision table
    - **How would a decision table be used?**

Decision table for the example

	A1B3	A1B4	A2B3	A2B4
A < B	T	T	F	F
C < D	T	F	T	F
A value	2	0	1	5
B value	5	1	0	2

Paths **A1B3** and **A2B4** give statement coverage
or Paths **A1B4** and **A2B3** give statement coverage

Selecting paths

- A program unit may contain a large number of paths.
  - Path selection becomes a problem
  - Some selected paths may be infeasible
- **What strategy would you use to select paths?**

Selecting paths – 2

- **What strategy would you use to select paths?**
  - Select as many short paths as possible
    - Tradeoffs?
  - Choose longer paths
    - Tradeoffs?

Selecting paths – 3

- **What about infeasible paths?**
  - What would you do about them?

Selecting paths – 4

- **What about infeasible paths?**
  - What would you do about them?
  - Make an effort to write program text with fewer or no infeasible paths.