Structural Based Testing Strategies

Control Flow Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Develop test cases to achieve control flow coverage

Code Coverage

- **It is important to analyze code coverage obtained by executing requirement's based test cases**
- **Code coverage can be assessed in terms of:**
  - Control flow
  - Data flow
- **Failure to obtain coverage may be due to:**
  - Undocumented requirements contained in the code
  - Dead code
  - Incomplete test cases for a requirement

Control Flow Coverage Levels

- Statement coverage
- Decision coverage
- Decision / Condition coverage
- Multiple condition coverage

Statement Coverage

- Develop test cases such that every statement is executed at least once.

```
if a < 10 or b > 5
  then
         x := 50
  else
         x := 0;
if w = 5 or y > 0
  then
         z := 48
  else
         z : = 5;
```

Decision Coverage

Develop test cases such that each branch is traversed at least once.

- What are examples of branch statements?
- Does decision coverage satisfy statement coverage?
- Does statement coverage satisfy decision coverage?

Decision / Condition Coverage

- Develop test cases such that each condition in a decision takes on all possible outcomes at least once and each decision takes on all possible outcomes at least once

Multiple Condition Coverage

- Develop test cases such that all combinations of conditions in a decision are tested

Binary Search Example

```
inputs:   table, num, key
outputs: found, loc
start := 1;
end := num;
found := false
while start <= end and not found
     middle := (start + end) / 2
     if key > table [middle]
          then start := middle + 1
          else if key = table [middle]
                    then found := true
                         loc := middle
                    else end := middle -1
```

Summary

Structural Based Testing Strategies

Data Flow Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Develop test cases to achieve data flow coverage

Approach

- Annotate control flow graph with 3 sets for each node
- Def(i) – set of variables defined in node I
- C-Use(i) – set of variables used in a computation in node I
- P-Use(i) – set of variables used in a test predicate

Example

```
Get x,z;
y := 0;
If x > 10
    then y := 15;
If z > 0
    then w := y+1
    else  w := y-1;
```

Definition Clear Path

- A definition clear path from node “i” to node “j” for a variable x is a path where x is defined in node j and either used in a test predicate or computation in node j and there is no re-definition of x between node i and node j

Example

```
get x,y;
a := 0;
b := 0;
if x > 10
    then w := a+1
           b := 4
    else  w := b+1
          a := 4;
If y > 10
    then   z := a+w
    else    z := b+w;
```

Definition Use (DU path) Coverage

- For each definition of a variable, develop test cases to execute all DU paths

- DU path starts with the definition of the variable and ends with either a computational or predicate use of the variable along a def-clear path

Example

```
get x,y;
a := 0;
b := 0;
if x > 10
    then w := a+1
         b := 4
    else  w := b+1
         a := 4;
If y > 10
    then   z := a+w
    else    z := b+w;
```

Summary

Structural Based Testing Strategies

Static Analysis

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Identify static analysis techniques

Data Flow Analysis

- **Model the flow of data in a program**
  - Where are variables defined
  - Where are variables used

- **Perform analysis without executing the program**

- **Look for data flow anomalies**

Example Data Flow Anomalies

- Variable defined and then redefined without being referenced
- Referencing an undefined variable
- Defining a variable but never using it
- Numerous tools available to perform anomaly detection

Huang’s Theorem

Let A,B,C be nonempty sets of character sequences whose smallest string is at least 1 character long. Let T be a 2-character string. Then if T is a substring of Ab^nC, then T will appear in AB^2C.

Summary

Structural Based Testing Strategies

Structured Testing

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Apply structured
testing technique

McCable Cyclomatic Complexity

- v(G) of a grap with e edges, n nodes and p connected components is e-n+p

- In a typical program:
  - v(G) = #test predicates +1

Example

```
S1;
if x < 10
    then S2
    else if y > 0
            then S3
            else S4;
If z = 5
    then S6
    else S7;
```

Application for Testing

- Impossible to test all paths through code
- Structured testing provides a strategy for testing a subset of paths
- Select a set of basis paths (number is $v(G)$)
- Linear combination of basis paths will generate any path
- Guarantees branch coverage

Identification of Basis Paths

- Select an arbitrary path through the graph as initial basis path
- Flip first decision while keeping other decisions constant
- Reset first decision and flip second decision
- Continue until all decisions have been flipped

Example

```
S1;
if x < 10
    then S2
    else if y > 0
            then S3
            else S4;
If z = 5
    then S6
    else S7;
```

Summary

Structural Based Testing Strategies

Symbolic Execution

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Utilize symbolic
execution

Symbolic Execution

- Technique for formally characterizing a path domain identifying a path condition
- All paths in the program form an execution tree
- Involves executing a program with symbolic values
- Identifies test data to execute a path or determination that a path is infeasible

Notation

- A variable "x" will have a succession of symbolic values: A_0, A_1, A_2 … as a path is traversed

- Subscripts refer to the number of the previous statement executed

Example

(0) input A,B
(1) A := A + B;
(2) B := A + B;
(3) A := 2 x A + B;
(4) C := A + 4;

Multiple Paths Example

```
if (x <= 0) or (y <= 0) then
(1)          x := x2;
             y := y2;
        else
(2)          x := x +1
             y := y + 1
        endif
        if (x < 1) or (y < 1) then
(3)          x := x + 1;
             y := y +1;
        else
(4)          x := x - 1;
             y := y – 1;
        endif
```

Example

Path Conditions

- In addition to symbolically evaluating a program variables along a path, we can also symbolically represent the conditions which are required for that path to be traversed

- The symbolic path condition must be expressed in terms of the initial symbolic values of the variables

Example for T,F Path

Example for T,F Path

Summary