Control Flow Testing

Contents

[Control Flow Testing 2](#_Toc443576415)

[Steps in Control Flow Testing 2](#_Toc443576416)

[Path Selection Criteria 2](#_Toc443576417)

[All-Path 2](#_Toc443576418)

[Statement Coverage 2](#_Toc443576419)

[Branch Coverage 2](#_Toc443576420)

[Predicate Coverage 3](#_Toc443576421)

[Control Flow Graph (CFG) 3](#_Toc443576422)

[Example of a CFG 3](#_Toc443576423)

[Selecting Paths 4](#_Toc443576424)

[All-Path Criteria 4](#_Toc443576425)

[Statement Coverage 5](#_Toc443576426)

[Predicate (Branch) Coverage 5](#_Toc443576427)

[Generating Test Input Data 5](#_Toc443576428)

[Terms 5](#_Toc443576429)

[Example 1 5](#_Toc443576430)

[Example 2 (infeasible path) 6](#_Toc443576431)

# Control Flow Testing

(Naik & Tripathy, 2008)

## Steps in Control Flow Testing

1. Select the path selection criteria and the SUT
2. Draw a control flow graph
3. Select paths using the path selection criteria
4. Generate test input data
5. If the paths are not feasible, go to 3
6. Perform testing

## Path Selection Criteria

### All-Path

A path is a sequence of steps through the program unit from the start to the end. The all-paths criteria selects all possible path through the program unit. Selecting all paths is ideal but there may a very large number of paths making all-path selection infeasible.

### Statement Coverage

Statement coverage refers the executing individual program statements and observing the outcome. If all statements are executed at least once then 100% statement coverage was achieved. Statement coverage is the weakest coverage criterion in program testing.

### Branch Coverage

A branch is an outgoing edge of the control flow graph. Process nodes have one branch while decision nodes have two branches.

|  |
| --- |
|  |

If all branches are executed at least once then 100% branch coverage was achieved.

### Predicate Coverage

Predicate coverage is a special form of branch coverage where each predicate is given its own decision node.

|  |
| --- |
|  |

## Control Flow Graph (CFG)

A CFG is a graphical representation of a program unit. Three symbols are used to construct the graph.

|  |
| --- |
|  |

Each computation and decision node is given a unique integer. Each branch from a decision node is labeled with a ‘T’ or an ‘F’.

### Example of a CFG

|  |
| --- |
| public static double ReturnAverage(int[] values, int AS, int MIN, int MAX)  {  int ti = 0, i = 0, tv = 0, sum = 0;  while(ti < AS && values[i] != 999)  {  ti++;  if(values[i] >= MIN && values[i] <= MAX)  {  tv++;  sum += values[i];  }  i++;  }  double average = 0;  if(tv > 0)  {  average = (double)sum / tv;  }  else  {  average = -999;  }  return average;  } |

|  |
| --- |
|  |

## Selecting Paths

### All-Path Criteria

**1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10-11(T)-13-14**

Others…

### Statement Coverage

**1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10-11(T)-13-14**

**1-2-3(T)-4(T)-5-6(T)-7(T)-8-9-3(F)-10-11(F)-12-14**

### Predicate (Branch) Coverage

**1-2-3(F)-10-11(F)-12-14**

**1-2-3(F)-10-11(T)-13-14**

Others…

## Generating Test Input Data

### Terms

**Input Vector**: An input vector is a collection of data values sent into the SUT.

**Predicate**: An expression that evaluates to either true or false.

**Path Predicate**: A set of predicates associated with a path.

**Predicate Interpretation**: Symbolically substituting operations along a path in order to express the predicates solely in terms of the input vector and constant values.

**Path Predicate Expression**: An interpreted path predicate.

### Example 1

**1-2-3(F)-10-11(F)-12-14**

**Predicate Interpretation Table**

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Interpretation** |
| **1** | <values, AS, MIN, MAX> |  |
| **2** | ti = 0, i = 0, tv = 0, sum = 0 |  |
| **3(F)** | ti < AS | 0 < AS |
| **10** | average = 0 |  |
| **11(F)** | tv > 0 | 0 > 0 |
| **12** | average = -999 |  |
| **14** | return average | return -999 |

**Path Predicate Expression**

|  |
| --- |
| 0 < AS ≡ false -- (1) |
| 0 > 0 ≡ false -- (2) |

**Input Vector (Test Data)**

|  |
| --- |
| values[0] = don’t care |
| AS = -1 |
| MIN = don’t care |
| MAX = don’t care |

### Example 2 (infeasible path)

**1-2-3(F)-10-11(T)-13-14**

**Predicate Interpretation Table**

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Interpretation** |
| **1** | <values, AS, MIN, MAX> |  |
| **2** | ti = 0, i = 0, tv = 0, sum = 0 |  |
| **3(F)** | ti < AS | 0 < AS |
| **10** | average = 0 |  |
| **11(T)** | tv > 0 | **0 > 0** (Infeasible path – cannot be true) |
| **13** | average = (double)sum/tv | **average = (double)0/0** |
| **14** | return average | **return (double)0/0** |