Test Adequacy Assessment using Control Flow and Data Flow

Data Flow based Adequacy

- CU: total number of c-uses in a program.
- PU: total number of p-uses.

Given a total of n variables $v_1, v_2, ..., v_n$ each defined at $d_i$ nodes.

$$CU = \sum_{i=1}^{m} \sum_{j=1}^{d_i} |dcu(v_i, j)|$$

$$PU = \sum_{i=1}^{n} \sum_{j=1}^{d_i} |dpu(v_i, j)|$$

c-use Coverage: Path Traversal

- Path (Start, .. q, k, .., z, .. End) covers the c-use at node z of x defined at node q, given that (k, ..., z) is def-clear with respect to x.

Exercise:
Devise a test that covers node 6 in dcu(z, 5):

t1: $<x=5, y=-1, count=1>$
Path = (1, 2, 5, 6, 7)
p-use Coverage

**P-use coverage:**

The p-use coverage of $T$ with respect to $(P, R)$ is computed as

$$\frac{PU}{(PU + PU_f)}$$

where $PU_C$ is the number of p-uses covered and $PU_f$ the number of infeasible p-uses. $T$ is considered adequate with respect to the p-use coverage criterion if its p-use coverage is 1.

Exercise:

Devise a test that covers the p-use of $y$ defined at node 6:

$t2: \langle x=-2, y=-1, \text{count}=3 \rangle$

Path = $\{1, 2, 3, 6, 2, 3, 4, 6, 2, 3, 6, 7\}$

t2 covers (3, 4) and (3, 6)

Exercise:

Is $T = \{t1, t2\}$ adequate w.r.t. to all-uses coverage for the running example program?

p-use Coverage: Path Traversal (cont.)

- A test covers p-use of $x$ at node $z$, if
  - Path (Start, ..., $q$, $k$, ..., $z$, $r$, ..., End) is a def-clear path, and
  - Path (Start, ..., $q$, $k$, ..., $z$, $s$, ..., End) is a def-clear path

All-uses Coverage

**All-uses coverage:**

The all-uses coverage of $T$ with respect to $(P, R)$ is computed as

$$\frac{CU + PU}{(CU + PU) - (CU_f + PU_f)}$$

where $CU$ is the total c-uses, $CU_C$ the number of c-uses covered, $PU_C$ the number of p-uses covered, $CU_f$ the number of infeasible c-uses and $PU_f$ the number of infeasible p-uses. $T$ is considered adequate with respect to the all-uses coverage criterion if its c-use coverage is 1.

Exercise:

Is $T = \{t1, t2\}$ adequate w.r.t. to all-uses coverage for the running example program?
Infeasible p- and c-uses

- Coverage of a c- or a p-use requires a path to be traversed through the program. However, if this path is infeasible, then some c- and p-uses that require this path to be traversed might also be infeasible.
- Infeasible uses are often difficult to determine without some hint from a test tool.
- Example
  - The c-use at node 4 of z defined at node 5

Other Data-Flow based Criteria

- There exist several other adequacy criteria based on data flows. Some of these are more powerful in their error-detection effectiveness than the c-, p-, and all-uses criteria.
- Examples
  - def-use chain or k-dr chain coverage. These are alternating sequences of def-use for one or more variables.
  - Data context and ordered data context coverage.

Infeasible p- and c-uses (cont.)

- For loop may lead to infeasible paths

```plaintext
begin
  int x, z, i
  input(x); z=0;
  for( i=1; i <= 5; i++ ){
    z=z+x*i;
  }
  output(x, z);
end
```
- Example path: (Start, 1, 2, 4, End)

Subsumes Relation

- Subsumes
  - Given a test set T that is adequate with respect to criterion C₁, what can we conclude about the adequacy of T with respect to another criterion C₂?
- Effectiveness
  - Given a test set T that is adequate with respect to criterion C, what can we expect regarding its effectiveness in revealing errors?
Structural and Functional Testing

- **Functional testing**
  - Compare program behavior against a requirement specification
- **Structural testing is:**
  - Functional testing with the addition of code-based adequacy measurement
  - Known as white-box testing
- Practitioners can perform structural testing as part of functional testing.

Summary

- We have reviewed the notion of test adequacy and enhancement.
- Two types of adequacy criteria considered
  - Control flow based: statement, decision, condition, multiple condition, MC/DC, and LCSAJ coverage. Many more exist.
  - Data flow based: c-use, p-uses, all-uses, k-dr chain, data context, elementary data context. Many more exist.
- Use of the criteria discussed requires a test tool that measures coverage during testing and displays it in a user-friendly manner.

Summary

- Often, it is believed that code coverage is useful at unit-level. Incremental and selective coverage measurement and enhancement of tests can allow the application of coverage-based testing to large programs.
- Adequacy to a coverage does not guarantee revealing all errors, but it is perhaps the most effective way to assess the amount of code that has been tested and what remains untested.
- Tests derived using black-box approaches can almost always be enhanced using one or more of the assessment criteria discussed.