Data Flow Testing

Chapter 10
Data Flow Testing

- Testing All-Nodes and All-Edges in a control flow graph may miss significant test cases.
- Testing All-Paths in a control flow graph is often too time-consuming.
- Can we select a subset of these paths that will reveal the most faults?
- Data Flow Testing focuses on the points at which variables receive values and the points at which these values are used.
Data Flow Analysis

- Can reveal interesting bugs
  - A variable that is defined but never used
  - A variable that is used but never defined
  - A variable that is defined twice before it is used
- Paths from the definition of a variable to its use are more likely to contain bugs
Definitions

- A node in the program graph is a **defining** node for variable $v$ if the value of $v$ is defined at the statement fragment in that node
  - Input, assignment, procedure calls

- A node in the program graph is a **usage** node for variable $v$ if the value of $v$ is used at the statement fragment in that node
  - Output, assignment, conditionals
More Definitions

- A usage node is a predicate use (P-Use) if variable $v$ appears in a predicate expression.
- A usage node is a computation use (C-Use) if variable $v$ appears in a computation.
- A definition-use path (du-path) with respect to a variable $v$ is a path whose first node is a defining node for $v$, and its last node is a usage node for $v$.
- A du-path with no other defining node for $v$ is a definition-clear path (dc-path).
```java
int max = 0;
int i = s.nextInt();
while (i > 0) {
    if (i > max) {
        max = i;
    }
    i = s.nextInt();
}
System.out.println(max);
```
int max = 0;
int i = s.nextInt();

while (i > 0)
    System.out.println(max);
    max = i;
    if (i > max)
        i = s.nextInt();
System.out.println(max);
du-paths in example

- Variable i
  - AB, AC, AD, EB, EC, ED

- Variable max
  - AF, AC, DC, DF
Data Flow Coverage Metrics

- Based on these definitions we can define a set of coverage metrics for a set of test cases

- We have already seen
  - All-Nodes
  - All-Edges
  - All-Paths
All-Defs Criterion

- For every variable $v$
- For every defining node $d$ of $v$
  - There exists at least one test case that follows a path from $d$ to a usage node of $v$
All-Uses Criterion

- For every variable v
- For every defining node d of v
- For every usage node u of v
  - There exists at least one test case that follows a path from d to u
All-P-Uses / Some-C-Uses

- For every variable v
- For every defining node d of v
- For every P-Use u of v
  - There exists at least one test case that follows a path from d to u
  - If there is no P-Use of v, there must exist at least one test case that follows a path from d to a C-Use of v
All-C-Uses / Some-P-Uses

For every variable $v$

For every defining node $d$ of $v$

For every C-Use $u$ of $v$

- There exists at least one test case that follows a path from $d$ to $u$

- If there is no C-Use of $v$, there must exist at least one test case that follows a path from $d$ to a P-Use of $v$
Data flow analysis issues

- Aliasing of variables causes serious problems!
- Working things out by hand for anything but small methods is hopeless
- Compiler-based tools help in determining coverage values