Software Reliability and Testing

Data Flow Coverage

Based on material by Professor Lori A. Clarke, University of Massachusetts.

Control Flow Coverage Criteria

- Statement coverage
- Branch coverage
- Path coverage

Can we also make use of data flow information to guide us in selecting test cases?

Path Selection Problem

- All branches: 1, 2, 4, 5, 7
- 1, 3, 4, 6, 7
- We haven’t exercised the relationship between the definition in statement 2 and the reference in statement 6?

Definitions

- \( d_n(x) \): variable \( x \) is assigned a value at node/statement \( n \).
- \( u_m(y) \): variable \( y \) is used at node/statement \( m \).
- A definition clear path \( p \) wrt \( x \) is a subpath where \( x \) is not defined at any of the nodes/statements in \( p \).
- A definition \( d_m(x) \) reaches a use \( u_n(x) \) iff there is a subpath \((m) • p • (n)\) such that \( p \) is a definition clear path wrt \( x \).
Data Flow Path Selection

- Rapps and Weyuker
  - Definition-clear subpaths from definitions to uses
- Laski and Korel
  - Combinations of definitions that reach uses at a node via a subpath

Rapps and Weyuker’s Criteria

- All-Defs
- All-Uses
- All-C-Uses, Some-P-Uses
- All-P-Uses, Some-C-Uses
- All-P-Uses
- All-Du-Paths

Rapps and Weyuker’s Criteria

- All-Defs
  - Some definition-clear subpath from each definition to some use reached by that definition
  - \( x = \ldots \)
  - \( \ldots = x \)
  - def-clear path
Rapps and Weyuker’s Data Flow Criteria

- **All-Uses**
  - Some definition-clear subpath from each definition to each use reached by that definition and each successor node of the use.

- **C-use** is a “computation use” E.g. \( y = x \times 2 \); P-use is a “predicate use” E.g. if \((x < 2)\) ...

- **All-C-Uses, Some-P-Uses**
  - Some definition-clear subpath from each definition to each C-Use reached by that definition.
  - If no C-Uses is reached by a definition, then some definition-clear subpath from that definition to at least one P-Use reached by that definition.

- **All-P-Uses, Some-C-Uses**
  - Some definition-clear subpath from each definition to each P-Use reached by that definition and each successor node of the use.
  - If no P-Uses is reached by a definition, then some definition-clear subpath from that definition to at least one C-Use reached by that definition.
Rapps and Weyuker’s Data Flow Criteria

- **All-DU-Paths (DU stands for definition use).**
  - All definition-clear subpaths that are cycle-free or simple-cycles from each definition to each use reached by that definition and each successor node of the use.

- \[ x \rightarrow \cdots \rightarrow \cdots \rightarrow \cdots \]

- **Example**

Example - All-Defs

- Requires: \( d_1(x) \) to a use
- Satisfactory path: 0, 1, 2, 4, 6

Example - All-Uses

- Requires:
  - \( d_1(x) \) to a \( u_2(x) \)
  - \( d_1(x) \) to a \( u_4(x) \)
  - \( d_1(x) \) to a \( u_5(x) \)
- Satisfactory path:
  - 0, 1, 2, 4, 5, 6
  - 0, 1, 3, 4, 6
Example - All-DU-Paths

- Requires:
  - \(d_1(x)\) to \(u_2(x)\)
  - \(d_1(x)\) to \(u_3(x)\)
  - Both paths for \(d_1(x)\) to \(u_5(x)\)

- Satisfactory path:
  - 0, 1, 2, 4, 5, 6
  - 0, 1, 3, 4, 5, 6

Laski and Korel’s Criteria

More Definitions

- Definition \(d_n(v)\) of variable \(v\) is said to be live at statement \(m\), if the definition at \(n\) reaches \(m\).
- \(DE(i)\) denotes the data environment for statement \(i\).
  - It is the set of all live definitions of all variables used (referenced) in statement \(i\).

More Definitions

- Data environment example:

```plaintext
1 read(x);    d1(x)
2 y = 4;      d2(y)
3 z = y;      d3(z)
4 while (x > 0) {
  5     if (x > y)
  6       y = y + 1; d6(y)
  else
  7     z = (y - x); d7(z)
}
8 write(z);   d8(z)
DE(3) = {d2(y)}
DE(4) = {d1(x), d2(y)}
DE(5) = {d1(x), d2(y), d3(z), d4(y)}
DE(6) = {d2(y), d6(y)}
DE(7) = {d1(x), d2(y), d3(z), d4(y)}
DE(8) = {d1(x), d2(y), d3(z), d4(y)}
DE(9) = {d1(x), d2(y)}
```
An elementary context of statement $i$ consists of the definitions that are live at statement $i$ for a particular path to statement $i$.

```
1  read(x) ;        d_1(x)  
2  y  = 1;          d_2(y)  
3  z  = y;          d_3(z)  
4  while (x > 0)    {   
5    if (x-y > 0)    
6      y  = y  + 1;  d_6(y)  
else                
7      x  = 1;      d_7(x)  
8    z  = y / x ;   d_8(z)  
}  
9  write(z) ;      
```

For example:

$$\text{ec}(8) = (d_3(x), d_2(y))$$

The data context of statement $i$, $DC(i)$, is defined as the set of all its elementary contexts.

```
1  read(x) ;        d_1(x)  
2  y  = 1;          d_2(y)  
3  z  = y;          d_3(z)  
4  while (x > 0)    {   
5    if (x-y > 0)    
6      y  = y  + 1;  d_6(y)  
else                
7      x  = 1;      d_7(x)  
8    z  = y / x ;   d_8(z)  
}  
9  write(z) ;      
```

$DC(3) = \{ d_2(y) \}$

$DC(4) = \{ d_1(x), d_7(x) \}$

$DC(5) = \{ d_1(x), d_2(y), d_7(x), d_7(y), d_3(x), d_4(y), d_3(x), d_5(y) \}$

$DC(6) = \{ d_2(y), d_6(y) \}$

$DC(8) = \{ d_1(x), d_7(x), d_3(x), d_2(y), d_7(x), d_6(y) \}$

$DC(9) = \{ d_3(z), d_8(z) \}$

An ordered data context also takes into account the order in which the definitions occur.

```
1  read(x) ;        d_1(x)  
2  y  = 1;          d_2(y)  
3  z  = y;          d_3(z)  
4  while (x > 0)    {   
5    if (x-y > 0)    
6      y  = y  + 1;  d_6(y)  
else                
7      x  = 1;      d_7(x)  
8    z  = y / x ;   d_8(z)  
}  
9  write(z) ;      
```

For example:

$$\text{DC}(8) = \{ (d_1(x), d_2(y)), (d_2(y), d_7(x)), (d_7(x), d_6(y)) \}$$

**Laski and Korel’s Criteria**

- **Context Coverage**
  - Every data context $DC(n)$ is exercised at least once.

- **Ordered Context Coverage**
  - Every ordered data context $ODC(n)$ is exercised at least once.
**Example - Context Coverage**

\[ DC(6) = \{ (d_1(x), d_4(y)), \]
\[ (d_3(x), d_1(y)), \]
\[ (d_3(x), d_4(y)) \} \]

Paths:
1, 2, 4, 5, 6
1, 2, 3, 5, 6
1, 2, 3, 5, 2, 4, 5, 6
Need to determine paths for all DC(6) to achieve context coverage.

**Example - Ordered Context Coverage**

\[ ODC(6) = \{ [d_1(x), d_4(y)], \]
\[ [d_3(x), d_1(y)], \]
\[ [d_3(x), d_4(y)], \]
\[ [d_4(y), d_3(x)] \} \]

Paths:
1, 2, 4, 5, 6
1, 2, 3, 5, 6
1, 2, 3, 5, 2, 4, 5, 6
1, 2, 4, 5, 2, 3, 5, 6
Need to determine paths for all ODC(6) to achieve ordered context coverage.

---

**How can we compare these criteria?**

- All select a set of paths, so compare the paths that they select
  - Set of paths that satisfy a criterion are not necessarily unique
  - E.g., \( s_1 \) or \( s_2 \) satisfies criterion A
  - \( s_1, s_2, \) or \( s_3 \) satisfies criterion B

---

**How can we compare these criteria?**

- Define a subsumption relationship
  - Criterion A subsumes criterion B iff for any flow graph:
    \[ P \text{ satisfies } A \implies P \text{ satisfies } B \]
    (P is a set of paths)
  - Criteria A is equivalent to criteria B iff A subsumes B, and B subsumes A.
Relationships among these criteria

- All-Paths
- Ordered Context Coverage
- Context Coverage
- All-C-Uses/Some-P-Uses
- All-Defs
- All-DU-Paths
- All-Uses
- All-P-Uses/Some-C-Uses
- All-P-Uses
- All-Edges
- All-Nodes

Conclusions
- An improvement over control flow techniques
- Provides a rationale for which combinations of subpaths to consider
- Most commonly used criteria is all-uses
- One problem with data flow coverage is infeasible paths
  - Don’t usually get 100% coverage