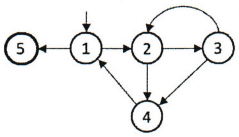
Exam Software Testing & Verification 2016/2017

22nd May 2017, 13:15–15:00, EDUC-THEATRON

Lecturer: Wishnu Prasetya

1 Part I (4pt)

1. What does it mean for a coverage criterion C_1 to subsumes another coverage criterion C_2?
2. What is a simple path and what is a prime path?
3. Consider the Control Flow Graph (CFG) below. Node-1 is the initial node, and node-5 is the exit node.



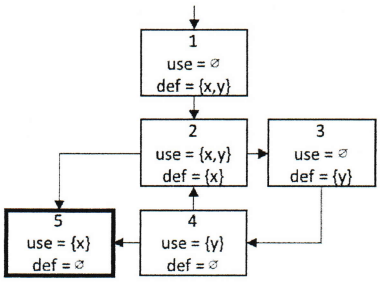
List all prime paths in this CFG, **that starts in node-2**.

4. Consider a program $P(x, y, z)$. The domains of x, y, and z are each partitioned into two blocks:

$$\begin{array}{lcl} x & \rightarrow & \{X_1, X_2\} \\ y & \rightarrow & \{Y_1, Y_2\} \\ z & \rightarrow & \{Z_1, Z_2\} \end{array}$$

Give a smallest **test set** that would give full Pair-wise Coverage (PWC) over those blocks.

5. Consider again the program P in No. 4. We select (X_1, Y_1, Z_1) as a so-called base test-case. Give the smallest **test set** that would give full Base Choice Coverage (BCC).
6. Give the definition of "du-path of a variable x".
7. Consider the following CFG of some program. The nodes are annotated with defs and uses information. If the same variable occurs in the def and the use of the same node n, we assume that within this node the def of this variable occurs *after* its use.



Which paths should be included as your test requirements (**TR**) to have full All-Defs Coverage (ADC)? Note that the program has two variables. Also note that you are asked to enumerate your TR, and *NOT* to specify a test set.

3. Consider the program fragment below. Assume that all array accesses are valid.

```
1  module A {
2    int[] zap(int N) {
3       int[] a = mkarray(...) ;
4       int[] b = mkarray(...) ;
5       int k = 0 ;
6       while (k<N) {
7          foo(a,k) ;  // call point (*)
8          if (b[k]>0)
9              b[k] = a[k]
10         k++ ;
11      }
12      return b ;
13   }
14 }
15
16 module B {
17  foo(int[] a, int k) {
18    if (k>9)
19      return ;
20    a[k] = a[k] + a[k+1] ;
21    if(a[k]<0) {
22      a[k] = 0 ;
23    }
24  }
25 }
```

The def and use of an array `a` is defined as follows:

- An assignment `a` =... is a def of `a`.
- An assignment `a`[e] = ... is a def of `a`.
- An expression `a`[e] in the guard of an **if** or a **while** is a use of `a`.
- An expression `a`[e] in the right-hand-side of an assignment is a use of `a`.

Furthermore:

- Passing a **variable** `v` directly in a program call such as `f(v)` does **not** in itself count as a use of `v`. The defs and uses of `v` inside f do count as defs and uses of `v`.

Your task: list all du-paths that should be included as test requirements (**TR**) for achieving full **All-coupling-use Coverage**. **For each path, mention the name of the coupling variable that the path is associated to.**

Use line numbers to describe your paths. For example: [2,3,4,5,6] describes the path from the start of `zap` up to its loop-head.

3 Part III (1pt)

1. Consider the following binding triples within `foo()`, and their corresponding coupling paths. In all cases, the context variable is o. All named paths are distinct.

binding triples			
coupling sequence	runtime type of context-var	coupling variables	coupling paths
C_1	A	$\{o.x\}$	σ_1
C_1	B	$\{o.y\}$	σ_2
C_2	A	$\{o.x, o.y\}$	τ_1 for $o.x$ τ_2 and τ_3 for $o.y$
C_2	B	$\{o.x\}$	τ_4

   (a) Give a minimal set of paths that should be included as the test requirements (**TR**) for achieving full All-Coupling-Defs-Uses (ACDU).

   (b) Give a minimal addition to the TR in (a) to specify the TR for achieving full All-Poly-Coupling-Defs-Uses (APCDU) coverage.

2. Consider a program $Q(s)$ that takes a string s as a parameter. The string has to follow the grammar shown below (described in the BNF notation):

$$
\begin{array}{lcll}
S & \rightarrow & \epsilon & \text{(RuleS1)} \\
S & \rightarrow & \text{"}x\text{"} & \text{(RuleS2)} \\
S & \rightarrow & B & \text{(RuleS3)} \\
B & \rightarrow & \text{"<" } S \text{ ">" } S & \text{(RuleB)}
\end{array}
$$

   Give a minimalistic test set for Q that would give full All Rule-Rule Coverage (ARRC) with respect to the above grammar. Only rule-combinations that are feasible need to be covered. We favor a larger test set with shorter test cases (and we do **not** favor a small test set with longer test cases).

   It may help (but you don't have to) if you present your test cases in the form of derivation trees.

8. Consider a program $Q(s)$ that takes a string s as a parameter. The string has to follow the grammar shown below (described in the BNF notation):

S	$\rightarrow$	ϵ	(RuleS1)
S	$\rightarrow$	A	(RuleS2)
S	$\rightarrow$	B	(RuleS3)
A	$\rightarrow$	"a" S "a" S	(RuleA)
B	$\rightarrow$	"b" S "b" S	(RuleB)

The non-terminals are S, A, B with S as the starting symbol. Symbols between quotes are terminals.

A tester wants to test the program Q using the string "$baabaa$" as the input. Prove that the string is a valid string in the above grammar. Prove this by showing a **derivation** that leads to this string.

9. Give the definition of "coupling du-path".

10. Name two uses of mutation testing.

2 Part II (5pt)

1. Consider a program with N parameters: $P(X_1, ..., X_N)$; N>0. The domain of X_N is divided into k blocks; we name them $B_1 ... B_k$. The domains of other parameters are left undivided; so each X_i with $i \neq N$ only has one block; we name it C_i.

   (a) Suppose we want to do pair-wise testing over those blocks. How many pairs of blocks have to be covered in total?

   (b) Give a minimal test set that would deliver full Pair-wise Coverage (PWC) over the blocks.

2. Prove that All-du-Paths Coverage (ADUPC) does **not** subsume Prime Path Coverage.