Software Testing, Quality Assurance and Maintenance Winter 2010

Lecture 9 — January 22, 2010

Patrick Lam *version 1*

So far, we've seen a number of coverage criteria for graphs, but I've been vague about how to actually construct graphs. For the most part, it's fairly obvious, and you've done it on assignments.

Graph Coverage for Source Code

Let's see formal definitions. Remember that we first defined the structural criteria: NC, EC, EPC, PPC, SPC, CPC. (Why are ADC, AUC, ADUPC, CRTC, SRTC inapplicable?)

Structural Graph Coverage for Source Code

Fundamental graph for source code: *Control-Flow Graph* (CFG).

- CFG nodes: zero or more statements;
- CFG edges: an edge (s_1, s_2) indicates that s_1 may be followed by s_2 in an execution.

Basic Blocks. We can simplify a CFG by grouping together statements which always execute together (in sequential programs):

```
     x = 5
     z = 2
l0:  if (z < 17) goto l1
     z = z + 1
     print (x)
     goto l0
l1:  nop
```

We use the following definition:

Definition 1 *A basic block has one entry point and one exit point.*

Note that a basic block may have multiple successors. However, there may not be any jumps into the middle of a basic block (which is why statement `l0` has its own basic block.)

0.1 Some Examples

We'll now see how to construct control-flow graph fragments for various program constructs.

if statements: The book puts the conditions (and hence uses) on the control-flow edges, rather than in the `if` node. I prefer putting the condition in the node.

```
if (z < 17)
  print (x);
else
  print (y);
```

```
if (z < 17)
  print(x);
}
```

(Recall that node coverage does not imply edge coverage.)

```
x = 0; y = 20;
while (x < y) {
  x ++; y ---;
}
```

Note that arbitrarily complicated structures may occur inside the loop body.

```
for (int i = 0; i < 57; i++) {
  if (i \% 3 == 0) {
    print (i);
  }
}
```

```
for (Widget w : l) {
  decorate(w);
}
```

```
switch (n) {
  case ‘I’: ...; break;
  case ‘J’: ...; /* fall through */
  case ‘K’: ...; break;
}
```

All of these graphs admit the notions of node coverage (statement coverage, basic block coverage) and edge coverage (branch coverage).