Software Testing, Quality Assurance & Maintenance (ECE453/CS447/CS647/SE465): Midterm

February 10, 2009

This open-book midterm has 5 questions and 90 points. Answer the questions in your answer book. You may consult any printed material (books, notes, etc).

Question 1 (20 points): First-uses

(10) In the context of a single method,

Criterion A. For each def-pair set $S = (n_i, n_j, v)$, TR contains all du-paths d in S.

Criterion B. For each def-pair set $S = (n_i, n_j, v)$, such that n_j is the first use in its basic block, TR contains all du-paths d in S.

Identify and explain the subsumption relationships between these criteria, giving examples as necessary.

(10) If n_i and n_j are in different methods, linked by a method call, define:

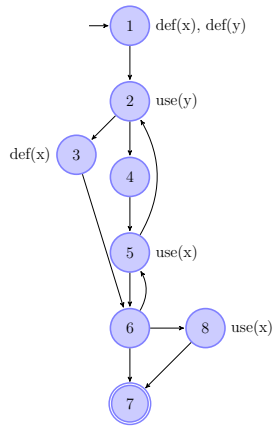
Criterion A'. For each def-pair set $S = (n_i, n_j, v)$, TR contains all du-paths d in S.

Criterion B'. For each def-pair set $S = (n_i, n_j, v)$, such that n_j is a first use in its method, TR contains all du-paths d in S.

Identify and explain the subsumption relationships between A' and B', giving examples as necessary.

Question 2 (20 points): Coverage Criteria

Here is a control-flow graph.



(a) Consider the set of test paths:

$$[1, 2, 3, 6, 8, 7], [1, 2, 4, 5, 6, 7], [1, 2, 4, 5, 2, 3, 6, 5, 6, 7].$$

Which of the following coverage criteria does the test set cover? Complete Path Coverage (CPC); Prime Path Coverage (PPC); Complete Round Trip Coverage (CRTC); Simple Round Trip Coverage (SRTC); Edge-Pair Coverage (EPC); Edge Coverage (EC); Node Coverage (NC); All-du-Paths-Coverage (ADUPC); All-Uses Coverage (AUC); All-Defs Coverage (ADC). (+1 for including or omitting each criterion correctly).

(b) What about this set of test paths:

$$[1, 2, 3, 6, 5, 2, 4, 5, 6, 7], [1, 2, 3, 6, 5, 2, 3, 6, 7], [1, 2, 4, 5, 2, 4, 5, 6, 7]$$

Question 3 (20 points): True or False

True or false:

1. Coverage criterion C_1 imposes an infeasible test requirement tr on a program P. Coverage criterion C_2 also imposes tr. There exists a test set that fully meets C_2.

2. Prime path coverage can always be satisfied with simple test paths.

3. A prime path in a graph is never a suffix of another prime path in the same graph.

4. Complete graph coverage is possible for all graphs.

5. A def-pair set is a union of def-path sets for that def[1].

6. Satisfying coupling inter-procedural data-flow coverage is always easier than satisfying full inter-procedural data-flow coverage.

7. An unreachable program fault can be detected using testing.

8. If test set T_1 achieves a higher coverage level than T_2 on a set of test requirements TR, then T_1 will detect more defects than T_2.

9. The control-flow graph corresponding to a Java method is sometimes not a Single-Entry/Single-Exit graph.

10. If test path p tours subpath q with sidetrips, then p also tours q with detours.

11. All-du-Paths is equivalent to All-Uses in all graphs without cycles.

12. Defs and uses for the same variable may appear on the same control-flow graph node.

13. A test case will always exclusively visit syntactically reachable nodes in a control-flow graph.

14. If p tours q, and q tours r with detours, then p also tours r with detours.

15. A basic block b may have two predecessors b_0 and b_1.

16. After executing a round-trip in a control-flow graph, the program's state is identical to its state preceding the round-trip.

17. Any path can be composed by concatenating prime paths.

18. Dead code makes it impossible to achieve node coverage.

19. A sidetrip is always a simple path.

[1]Don't believe everything you read in textbooks!

20. Fewer paths are semantically possible through a program than are syntactically reachable.

21. It is conceptually possible to cover all-du-paths for a shared-data coupling variable.

Question 4 (10 points): AUC vs EC

We've stated that All-Uses Coverage subsumes Edge Coverage. (1) Under what conditions does this subsumption relationship hold? (9) Give an example where AUC does not subsume EC; you may falsify the conditions you stated in the first part.

Question 5 (20 points): Control-Flow Graphs

Create a control-flow graph of the following method (5 points) and provide inputs that achieve edge coverage (7 points) and all-defs coverage (7 points) on your control-flow graph. (At conditionals, you may put uses either on the edges or on the nodes.)

We consider `x.put()` to be a use of `x`. (1 point) What type of coupling is going on with `rhsToContainingStatement`?

Examples of statements `s` are `x = 5, x = y + z, x = new Foo()` (all `AssignStmts`) and `return;`. You may assume that "5" is not an `Expr`.

```
public void Foo(LinkedList units)
{
    rhsToContainingStmt = new HashMap<Value, Unit>();
    emptySet = new ToppedSet(new ArraySparseSet());

    unitToGenerateSet = new HashMap();
    Iterator unitIt = units.iterator();

    while(unitIt.hasNext())
    {
      Unit s = (Unit) unitIt.next();

      FlowSet genSet = emptySet.clone();
```

```
        if (s instanceof AssignStmt)
        {
          AssignStmt as = (AssignStmt)s;
          if (as.getRightOp() instanceof Expr)
          {
            Value gen = as.getRightOp();
            rhsToContainingStmt.put(gen, s);

            if (gen instanceof NewExpr)
                break;
            genSet.add(gen, genSet);
          }
        }

        unitToGenerateSet.put(s, genSet);
      }
}
```