Software Testing, Quality Assurance and Maintenance Winter 2010

Lecture 19 — February 22, 2010

Patrick Lam *version 2*

Infeasibility and Subsumption

Of course, some test requirements are infeasible. The ICC criteria (not seen this year) are often infeasible, and it's easy to make RACC infeasible as well.

Workaround I. Satisfy feasible TRs, drop infeasible TRs.

Workaround II. If you can't satisfy a particular coverage criterion, use a looser criterion. For instance, if you can't satisfy RACC, settle for CACC.

Workaround II is analogous to best-effort touring in the graph coverage case.

Causes of infeasibility. In response to a question in class, I described possible causes of infeasibility. From the definition, a logic coverage test requirement t is infeasible if no test case can satisfy test requirement t. Here's an example of a case where predicate coverage is infeasible:

```
void m(char[] c) {
  if (c.length < 0) {
    print ("infeasible");
 }
}
```

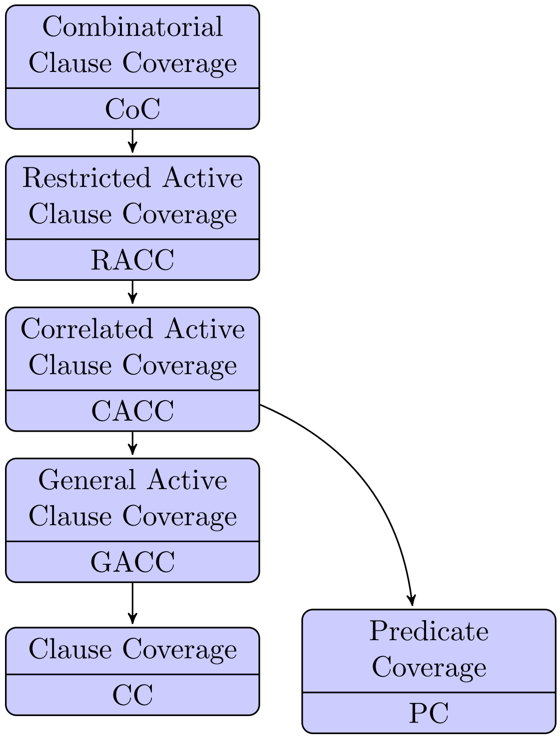
Clearly, no array c can have negative length, so the predicate can't be false.

I can think of the following ways for a coverage test requirement to be infeasible; these ways reflect the steps you need to follow to satisfy test requirements.

- The predicate is unreachable in its context (as in graph reachability); or
- The method containing the predicate never assigns appropriate values to its variables to cause the desired clause values (as in the example above); or
- A clause never determines a predicate and you are trying to achieve an active clause coverage criterion.

Subsumption Chart

Without inactive clause criteria, we can simplify the subsumption chart in the book to:



How to get Logic Coverage

To achieve logic coverage:

- identify the predicates $p \in P$ in the program fragment under test;
- figure out how to reach each of the predicates;
- make c determine p (for the active clause criteria); and
- find values for (program) variables to meet various criteria.

We've seen how to make c determine p; let's look at an example where we find values for variables to meet the criteria.

Predicates. We'll use the following predicate, modified from the textbook example `TestPat`:

```
isPat == false && iSub + pL - 1 < sL || subject[iSub] == pattern[0]
```

We can assign names to these clauses, giving the symbolic predicate:

Reaching a Predicate. A CFG-based example:

Determination. We analyze the predicate and determine the conditions under which each clause determines the predicate:

Mapping Values to Predicates. Now we need to find program values that correspond to different predicate values. `isPat` is a boolean variable, so it's easy. The code says that `pL` and `sL` are lengths of argument arrays, so we need to assign arrays such that `iSub + pL - 1 < sL`. Finally, we can assign values to `iSub` and `subject` such that `subject[iSub]` is equal to or different from `pattern`, as needed. Examples:

Coverage Criteria. We now list the coverage criteria. For PC, we have:

CC:

CoC:

GACC:

(What are all of the ways to satisfy GACC?)

CACC: additional constraint that each pair must cause p to be both true and false.

RACC:

Concluding Comments on Logic Criteria. GACC doesn't imply predicate coverage, but often does when 3 or more terms are involved. RACC can be unsatisfiable. CACC seems to often be "just right".