Software Testing, Quality Assurance and Maintenance Winter 2010

Lecture 16 — February 8, 2010

Patrick Lam *version 2*

A Heuristic: Active Clauses

So, in general, we can't evaluate every value of every clause, because there are too many of them. The next best thing is to focus on each clause and make sure it affects the predicate. That is, we'll test each clause while it is *active*.

Example. Consider the following clause:

$$p = x \wedge y \vee (z \wedge w)$$

Let's say that we focus on y; we'll call it the major clause. (We may designate any clause as a major clause at our whim.) That makes x, z, w minor clauses.

We want to make y *determine* p with certain minor clause values. That is:

- if we set y to true, then p evaluates to some value X;
- if we set y to false, then p must evaluate to $\neg X$.

The truth assignment:

$$x = \textvisiblespace \quad z = \textvisiblespace \quad w = \textvisiblespace$$

will make y determine p; in particular, y true makes p true and t false makes p false.

Definition 1 *Designate c_i as a major clause in predicate p. Then c_i* determines *p if the minor clauses $c_j \in p, j \neq i$, have values such that changing the truth value of c_i changes the truth value of p.*

We do *not* require that c_i has the same truth value as p. That requirement leads to trouble e.g. for the predicate:

Informally, determination tests each clause in a context where that clause has an effect.

Example.

$$p = a \vee b$$

Note that b does not determine p when:

That is, testing b has no effect; the test set { true, false } does not test a or b effectively.

Here is a variant of clause coverage which uses determination.

Definition 2 *Active Clause Coverage (ACC). For each $p \in P$ and making each clause $c_i \in C_p$ major, choose assignments for minor clauses $c_j, j \neq i$ such that c_i determines p. TR has two requirements for each c_i: c_i evaluates to true and c_i evaluates to false.*

This definition is somewhat ambiguous. We'll refine it soon.

Example. For $p = a \vee b$, make a major. We need b to be false for a to determine p. This leads to the TRs:

and similarly for b to determine p we need TRs:

Note the overlap between test requirements; it will always exist, meaning that our set of TRs for active clause coverage are:

In general, for a predicate with n clauses, we need $n+1$ test requirements. (It might seem that we'd need $2n$ clauses, but we don't. Why?)