Software Testing, Quality Assurance and Maintenance Winter 2015

Lecture 32 — March 25, 2015

Patrick Lam *version 0*

Combining Characteristics

We next investigate how to test multiple inputs/characteristics, each with their own partitions.

Consider three inputs, with the following representative values from each block of a partition:

$$[A, B] \qquad [1, 2, 3] \qquad [x, y]$$

Then the obvious exhaustive coverage criterion would be:

Criterion 1 All Combinations Coverage *(ACoC). TR contains all combinations of blocks from all characteristics.*

and we could start enumerating test cases: $(A, 1, x), (A, 1, y), (A, 2, x), \ldots$; there would be 12 tests in general, because the number of test requirements is the product

$$\prod^{Q} B_i$$

This can get too big if there are too many characteristics, so we'll propose heuristics. For instance, we might try each block at least once.

Criterion 2 Each Clause Coverage *(ECC). TR contains the requirement to include one value from each block for each characteristic in some test case.*

The following test suite would work: $(A, 1, x), (B, 2, y), (A, 3, x)$. This criterion is analogous to clause coverage, and requires at least as many tests as the largest number of blocks in a partition.

On the TriTyp interface-based IDM, we can satisfy ECC with four tests: $(2, 2, 2), (1, 1, 1), (0, 0, 0)$ and $(-1, -1, -1)$; note that these tests don't tell you much.

Combining Values. Imposing more test requirements might require more interesting test cases.

Criterion 3 Pair-Wise Coverage *(PWC). TR contains the requirement to combine a value for each block for each characteristic with some value from every other block for other characteristics.*

PWC imposes 16 test requirements for our running example:

$$(A,1),(A,2),(A,3),(A,x),(A,y),(B,1),(B,2),\ldots,(3,y),$$

and we can satisfy these requirements with 8 test cases:

$$(A,1,x),(A,2,x),(A,3,x),(A,*,y),(B,1,y),(B,2,y),(B,3,y),(B,*,x).$$

(The book contains a bogus analysis of the test set size; it actually analyzes # of test requirements.)

Criterion 4 T-Wise Coverage *(TWC). TR contains the requirement to combine a value for each block for each characteristic with T values from other blocks for other characteristics.*

When T is the number of partitions, TWC is the same as ACoC. $T > 2$ doesn't seem to help much.

Base Choice Coverage. We've treated all blocks as equivalent so far. We'll now pick one block as the most important block and call it the base choice. Examples: simplest, smallest, first (with respect to some ordering), most likely.

Criterion 5 Base Choice Coverage *(BCC). Choose a base choice block for each characteristic, and a base test by combining the base choices for all characteristics. For each characteristic c, TR contains the requirement for a test which varies the base test by using all other blocks for characteristic c.*

In our example, suppose the base choice blocks are A, 1 and x. Then the base choice test is $(A,1,x)$. Other tests would be $(\mathbf{B},1,x),(A,\mathbf{2},x),(A,\mathbf{3},x),(A,1,\mathbf{y})$.

Number of tests:

$$1+\sum^{Q}(B_i-1)$$

where B_i is the number of blocks for characteristic i.

For TriTyp, we have $1+3+3+3=10$ tests. Say that the block ">1" is the base choice. A base choice could be $(2,2,2)$ and the non-base choices might be:

$$(2,2,\mathbf{1}),(2,2,\mathbf{0}),(2,2,-\mathbf{1}),(2,\mathbf{1},2),(2,\mathbf{0},2),(2,-\mathbf{1},2),(\mathbf{1},2,2),(\mathbf{0},2,2),(-\mathbf{1},2,2).$$

BCC is like ECC except that you're only allowed to change 1 thing at a time from the base. We can also allow more than 1 base choice:

Criterion 6 Multiple Base Choice Coverage *(MBCC). Choose one or more base choice blocks for each characteristic, and form base tests by using each base choices for each characteristic at least once. For each characteristic c, TR contains the requirement for a test which varies each base test by using all other blocks for characteristic c.*

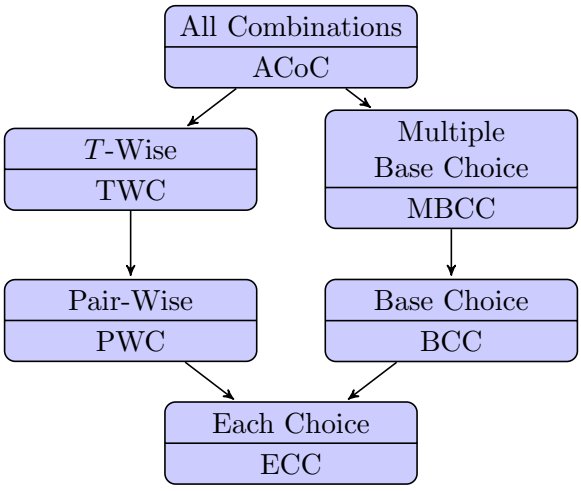
For example, we could have two base choices for side 1 in TriTyp, >1 and $=1$. This gives the base tests $(2,2,2)$ and $(1,2,2)$. Then we proceed as in plain BCC, but using each of the two base tests.

Upper bound on number of tests:

$$M+\sum^{Q}M(B_i-m_i)$$

where M is the number of base tests and M_i is the number of base choices for characteristic i.

Subsumption Chart



Constraints Among Partitions

Picking choices can lead to infeasible test requirements. For example, it's impossible to find test cases for scalene triangles with negative side lengths, because the negative side length forces the classification "invalid". Similarly, "`containsElement` returns true" requires the input to satisfy "element non-null".

Handling infeasible TRs. We can simply drop infeasible TRs, for ACoC, PWC, and TWC. For BCC, we can change the base case, e.g. if we choose base case "scalene", then we can't test $S1 < 0$. Instead, we can add an additional base case or modify the base case to "invalid".

Sort example. A sort routine might take an array and could return a sorted array, the largest element and the smallest element.

Possible characteristics:

Potential partitioning:

Not all combinations are possible.