Combinatorial Coverage Measurement Tool

User Guide

January 26, 2011

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

Text derived from NIST SP 800-142, Practical Combinatorial Testing, Oct. 2010.

Contact: Rick Kuhn; kuhn@nist.gov

Combinatorial Coverage Measurement

Since it is nearly always intractable to test all possible combinations, combinatorial testing is a reasonable alternative. For some value of *t*, testing all *t*-way interactions among *n* parameters will detect nearly all errors. It is possible that $t = n$, but reviewing the empirical data on failures, we would expect *t* to be relatively small. Determining the level of input or configuration state space coverage can help in understanding the degree of risk that remains after testing. If 90% - 100% of the state space has been covered, then presumably the risk is small, but if coverage is much smaller, then the risk may be substantial. This tutorial describes some measures of combinatorial coverage that can be helpful in estimating this risk.

Even in the absence of knowledge about a program's inner structure, we can apply combinatorial methods to produce precise and useful measures. In this case, we measure the state space of inputs. Suppose we have a program that accepts two inputs, *x* and *y*, with 10 values each. Then the input state space consists of the $10^2 = 100$ pairs of *x* and *y* values, which can be pictured as a checkerboard square of 10 rows by 10 columns. With three inputs, *x*, *y*, and *z*, we would have a cube with $10^3 = 1{,}000$ points in its input state space. Extending the example to *n* inputs we would have a (hard to visualize) hypercube of *n* dimensions with 10^n points. Exhaustive testing would require inputs of all 10^n combinations, but combinatorial testing could be used to reduce the size of the test set.

How should state space coverage be measured? Looking closely at the nature of combinatorial testing leads to several measures that are useful. We begin by introducing what will be called a *variable-value configuration.*

Definition. For a set of *t* variables, a variable-value configuration is a set of *t* valid values, one for each of the variables.

Example. Given four binary variables, *a, b, c*, and *d*, *a*=0, *c*=1, *d*=0 is a variable-value configuration, and *a*=1, *c*=1, *d*=0 is a different variable-value configuration for the same three variables *a, c*, and *d*.

Simple *t* -way combination coverage

Of the total number of *t*-way combinations for a given collection of variables, what percentage will be covered by the test set? If the test set is a covering array, then coverage is 100%, by definition, but many test sets not based on covering arrays may still provide significant *t*-way coverage. If the test set is large, but not designed as a covering array, it is very possible that it provides 2-way coverage or better. For example, random input generation may have been used to produce the tests, and good branch or condition coverage may have been achieved. In addition to the structural coverage figure, for software assurance it would be helpful to know what percentage of 2-way, 3-way, etc. coverage has been obtained.

Definition: For a given test set for *n* variables, simple *t*-way combination coverage is the proportion of *t*-way combinations of *n* variables for which all variable-values configurations are fully covered.

Example. Figure 1 shows an example with four binary variables, *a, b, c*, and *d*, where each row represents a test. Of the six 2-way combinations, *ab, ac, ad, bc, bd, cd*, only *bd* and *cd* have all four binary values covered, so simple 2-way coverage for the four tests in Figure 1 is 1/3 = 33.3%. There are four 3-way combinations, *abc, abd, acd, bcd*, each with eight possible configurations: 000, 001, 010, 011, 100, 101, 110, 111. Of the four combinations, none has all eight configurations covered, so simple 3-way coverage for this test set is 0%.

a	b	c	d
0	0	0	0
0	1	1	0
1	0	0	1
0	1	1	1

Figure 1. An example test array for a system with four binary components

(*t* + *k*)-way combination coverage

A test set that provides full combinatorial coverage for *t*-way combinations will also provide some degree of coverage for (*t*+1)-way combinations, (*t*+2)-way combinations, etc. This statistic may be useful for comparing two combinatorial test sets. For example, different algorithms may be used to generate 3-way covering arrays. They both achieve 100% 3-way coverage, but if one provides better 4-way and 5-way coverage, then it can be considered to provide more software testing assurance.

Definition. For a given test set for *n* variables, (*t*+*k*)-way combination coverage is the proportion of (*t*+*k*)-way combinations of *n* variables for which all variable-values configurations are fully covered. (Note that this measure would normally be applied only to a *t*-way covering array, as a measure of coverage beyond *t*).

Example. If the test set in Figure 1 is extended as shown in Figure 2, we can extend 3-way coverage. For this test set, *bcd* is covered, out of the four 3-way combinations, so 2-way coverage is 100%, and (2+1)-way = 3-way coverage is 25%.

a	b	c	d
0	0	0	0
0	1	1	0
1	0	0	1
0	1	1	1
0	1	0	1
1	0	1	1
1	0	1	0
0	1	0	0

Figure 2. Eight tests for four binary variables.

Variable-Value Configuration coverage

So far we have only considered measures of the proportion of combinations for which all configurations of *t* variables are fully covered. But when *t* variables with *v* values each are considered, each *t*-tuple has v^t configurations. For example, in pairwise (2-way) coverage of binary variables, every 2-way combination has $2^2 = 4$ configurations: 00, 01, 10, 11. We can define two measures with respect to configurations:

Definition. For a given combination of *t* variables, variable-value configuration coverage is the proportion of variable-value configurations that are covered.

Definition. For a given set of *n* variables, (*p*, *t*)-completeness is the proportion of the C(*n*, *t*) combinations that have configuration coverage of at least *p*.

Example. For Figure 2 above, there are C(4, 2) = 6 possible variable combinations and $C(4,2)2^2 = 24$ possible variable-value configurations. Of these, 19 variable-value configurations are covered and the only ones missing are *ab*=11, *ac*=11, *ad*=10, *bc*=01, *bc*=10. But only two, *bd* and *cd*, are covered with all 4 value pairs. So for the basic definition of simple *t*-way coverage, we have only 33% (2/6) coverage, but 79% (19/24) for the configuration coverage metric. For a better understanding of this test set, we can compute the configuration coverage for each of the six variable combinations, as shown in Figure 3. So for this test set, one of the combinations (*bc*) is covered at the 50% level, three (*ab*, *ac*, *ad*) are covered at the 75% level, and two (*bd*, *cd*) are covered at the 100% level. And, as noted above, for the whole set of tests, 79% of variable-value configurations are covered. All 2-way combinations have at least 50% configuration coverage, so (.50, 2)-completeness for this set of tests is 100%.

Although the example in Figure 3 uses variables with the same number of values, this is not essential for the measurement. Coverage measurement tools that we have developed compute coverage for test sets in which parameters have differing numbers of values, as shown in Figure 4 and Figure 5.

Vars	Configurations covered	Config coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

- *total 2-way cove*rage = 19/24 = .79167
- *(.50, 2)-completeness* = 6/6 = 1.0
- *(.75, 2)-completeness* = 5/6 = 0.83333
- *(1.0, 2)-completeness* = 2/6 = 0.33333

Figure 3. The test array covers all possible 2-way combinations of a, b, c, and d to different levels.

Figure 4 is an example of coverage for a $2^{87}3^2 4^5$ set (87 binary, two 3-value, and five 4-value) of input variables (blue=2-way, pink=3-way, yellow=4-way). This particular test set was not a covering array, but pairwise coverage is still quite good, with about 95% of the variables having all possible 2-way configurations covered. Even for 4-way combinations we see that all variables have at least 28% of their configurations covered, and about 25% of them have about 98% or more of 4-way configurations covered. Figure 5 shows a similar plot for a $2^{79}3^1 4^1 6^1 9^1$ configuration.

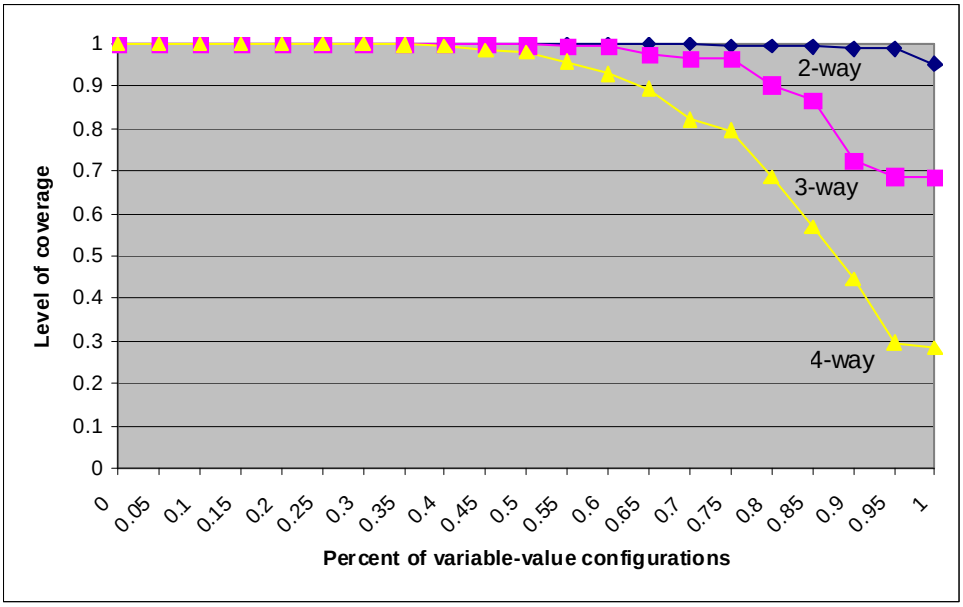

Figure 4. Configuration coverage for $2^{87}3^{2}4^{5}$ inputs.

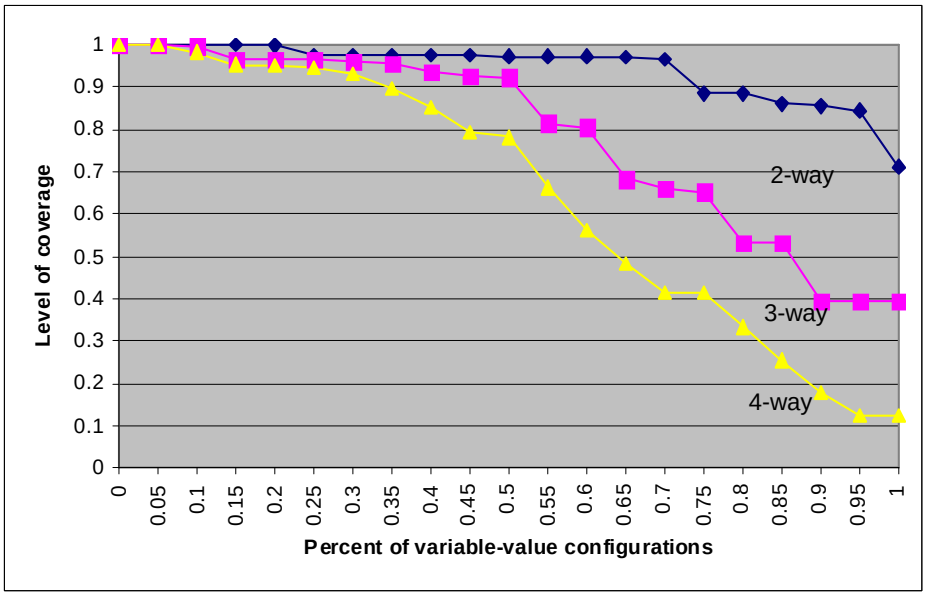

Figure 5. Configuration coverage for $2^{79}3^{1}4^{1}6^{1}9^{1}$ inputs.

Using Coverage Measurement Tool

The tool (Figure 6) implements the coverage measures documented in Chapter 6 of NIST SP 800-142.

1. Input

The input file must be a comma-separated values format file where each row represents a test and each column represents a parameter.

2. Modes of operation

The tool has two modes of operation: fully automatic, and user specified.

Full automatic

This mode is entered by checking "*Auto-detect N tests, N params*" in the upper left corner. It may be used when the following conditions are met:

1. All parameters have discrete values. Values may be numeric or alphanumeric.
2. All parameters have 10 or fewer values.

To use this mode, click on the checkbox, then click on "*Load input file*" and select the test file to be analyzed. After it loads, coverage measures may be computed by clicking on the appropriate button.

User specified values

This mode is entered by unchecking "*Auto-detect N tests, N params*" in the upper left corner. The number of tests and number of parameters should then be specified using the numeric fields on the left. To indicate values of each parameter, two choices are possible:

1. Detect all values automatically. This mode may be used when the conditions for parameters required by *Full Automatic* mode are met. (In this case, the only user-specified values are number of tests and parameters. This may be useful when the user wishes to load only part of a test file.
2. Set range boundaries. To use this mode, the user must specify the values of each parameter. Parameters are numbered 0 through n-1. For each parameter, the user indicates the values with one of two choices:
    a. Tell the system to detect values automatically, provided the parameter has 10 or fewer discrete values. This choice is indicated by clicking "*Detect*". The system then advances to the next parameter.
    b. For continuous-valued parameters, specify equivalence classes by indicating the number of value classes (2 to 10), and boundaries between the classes. This choice is indicated by setting the appropriate number of values, then clicking "*Set*", then specifying the boundaries. To set boundaries, enter the numeric value of the boundary and click "*Save bound*". The system will advance to the next boundary to be entered until all have been completed. Boundaries may include decimal values.

When all parameter values have been entered, they may be reviewed by using the "*Prev*" and "*Next*" buttons. Values for the parameter, or range boundaries, are displayed in the window labeled "*Values for this parameter*".

Charts

Charts display coverage measurements as documented in Chapter 6 of SP 800-142. Charts may be saved to an image file using "*Save chart*".

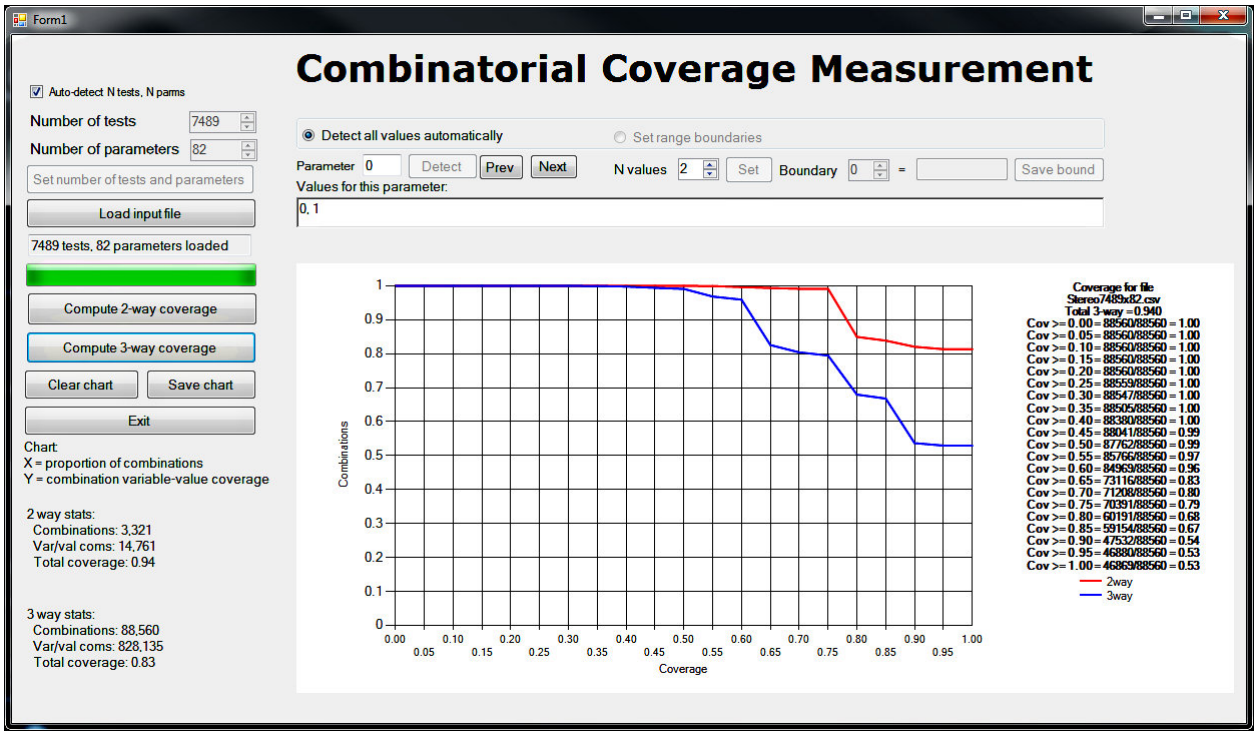

Figure 6. Coverage Measurement Tool user interface.

Example 1: Click "*Auto-detect N tests, N params*", then "*Load input file*". Select test18x8. The file will load, then click "*Compute 2-way coverage*" and/or "*Compute 3-way coverage*". Coverage will be shown as 100% for 100% of combinations for both 2-way and 3-way. Next, load test10x8 and compute coverage. Coverage will be shown as 96% for 2-way and 74% for 3-way. Coverage curves will show that for 2-way, 75% of combinations are covered at the 100% level, and all combinations are covered at 82% or above. For 3-way coverage, the situation is more complex. 60% of combinations are covered at the 100% level, some at the 70% level, some at 23%, and some at only 2%.

Example 2: Leaving "*Auto-detect*" checked, load rang18x8, then click "*Compute 2-way coverage*". The system will indicate an error, because parameter 2 has range values that cannot be detected automatically.

Example 3: Now specify values for parameter 2. Uncheck "*Auto-detect*" and then set the number of tests as 18 and number of parameters as 8. Click "*Set number of tests and parameters*". Then click "*Set range boundaries*". For parameter 0, click "*Detect*", and do the same for parameter 1. For parameter 2, set "*N values*" to 2 (the number of ranges) and click "Set". Then set boundary 0 as 50 and click "*Save bound*". Because there is only one boundary needed for 2 ranges, the system will advance to parameter 3. Click "*Detect*" for this one and all remaining parameters. After values and ranges have been set, they may be reviewed using the "*Prev*" and "*Next*" buttons. If all are correct, coverage may then be computed. For 2-way, all combinations are covered at the 100% level and for 3-way, all are covered at 93% or above and 85% are covered at the 100% level.

Example 4: Load stereo7489x82 and specify "*Auto-detect*". This large file should load quickly but coverage calculation may take several seconds for 2-way and a minute or more for 3-way coverage, depending on the computer used.

Tool Features and Options

Auto-detect N tests, N params – checkbox to specify if values should be detected are will be set by user. *Set range boundaries* is disabled when auto-detect is selected.

Number of tests – to set number of rows (tests) in input file, if known.
Number of parameters – this must also be set by the user when *Number of tests* is set.

Set number of tests and parameters – click this button after setting the number of tests and parameters.

Detect all values automatically – selecting this causes all values to be detected in the same manner as when *Auto-detect N tests, N params* is selected.

Set range boundaries – Setting this option enables the row of controls directly below it. Operation of these controls is as follows:

> *Parameter*: displays the current parameter, numbered 0 through *Number of parameters* – 1
>
> Detect: indicates that the value for this parameter is to be detected automatically
>
> *Prev*/*Next*: moves to the previous/next parameter in sequence
>
> *N values*: the number of equivalence class values that will be used for this parameter. For example, a parameter percent, with possible values 0.0 to 100.0 may be divided into quarters 0..25.0, 25.1..50.0, 50.1..75.0, and 75.1..100, so *N values* would be set to 4 to indicate the four equivalence classes.
>
> *Set*: must be clicked after N values is set, to set the number of values for this class.
>
> *Boundary*: indicates the current equivalence class boundary, numbered 0 through *N values* – 2. For example, in the percent example with four equivalence classes, there are three boundaries.
>
> *Save bound*: sets the boundary value entered in the box to the left of this control. values may be entered with decimals.

Compute 2-*way*/3-*way coverage*: calculates coverage.

Clear chart: removes any currently-displayed graph.

Save chart: opens a dialog box to allow user to save the displayed graph as an image file.

Exit: terminates the program. Charts are not automatically saved, and should be stored first using *Save chart.*