Test Generation from Combinatorial Designs

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Jonsson School of Engineering and Computer Science

Topics: Software Testing

Part II: Test Generation

3. Domain Partitioning

4. Predicate Analysis

5. Test Generation from Finite State Models

6. Test Generation from Combinatorial Designs

Test Configuration

Combinations of factors such as the operating system, network connection, and hardware platform, lead to a variety of operational environments.

Each environment corresponds to a given set of values for each factor, known as a test configuration.

- **Windows XP, dial-up connection, and a PC with 512MB of main memory, is one possible configuration**

For high reliability, the application must be tested under as many test configurations, or environments, as possible.

Test Configuration vs Test Set

A test configuration is a combination of factors corresponding to hardware and software within which an application is to operate.

A test set is a collection of test cases.

- **each test case consists of input values and expected output**

Techniques useful in deriving test configurations are useful for deriving test sets… and vice versa.

Input and Configuration Space

The input space of a program P consists of k-tuples of values that could be input to P during execution.

- **Consider program P that takes two integers x>0 and y>0 as inputs.**
  - **The input space of P is the set of all pairs of positive non-zero integers.**

The configuration space of P consists of all possible settings of the environment variables under which P could be used.

Configuration Space Example

Suppose that program P is intended to be executed

- **under the Windows or MacOS operating systems**
- **through the Netscape or Safari browsers**
- **must be able to print to a local or a networked printer**

The configuration space of P consists of triples (X, Y, Z) where

- **X represents an operating system**
- **Y a browser**
- **Z a local or a networked printer**

Factors and Levels

Consider a program P that takes n inputs corresponding to variables X_1, X_2, … X_n

- **refer to the inputs as factors**
- **also referred to as test parameters or values**

Assume that each factor may be set at any one from a total of ci, 1≤i≤ n values.

- **each value assignable to a factor is known as a level**
- **|F| refers to the number of levels for factor F**

Factor Combinations

A set of values, one for each factor, is known as a factor combination.

Suppose that program P has two input variables X and Y.

- **suppose that X and Y may each assume a value from the set {a, b, c} and {d, e, f} respectively**

P has 2 factors and 3 levels for each factor.

- **a total of $3^2 = 9$ factor combinations**
  - **{(a,d), (a,e), (a,f), (b,d), (b,e), (b,f), (c,d), (c,e), (c,f)}**

Too Many Factor Combinations?

In general, for k factors with each factor assuming a value from a set of n values, the total number of factor combinations is n^k.

Suppose that each factor combination yields one test case.

- **exhaustive testing could be exorbitantly large**
- **if a program has 15 factors with 4 levels each, the total number of tests is $4^{15} \sim 10^9$**
- **executing a billion tests might be impractical…**

Mathur, Example 6.4

The sort utility has several options and makes an interesting example for the identification of factors and levels.

The command line for sort is

sort [-cmu] [-o output] [-T directory] [-y [kmem]] [-z recsz] [-dfiMnr] [-b] [t char] [-k keydef] [+pos1[-pos2]] [file...]

20 factors for the sort command

- **approximately 1.9×10^9 combinations**

Factor	Meaning	Levels			
-	Forces the source to be the standard input	Unused	Used		
-c	Verify the the input is sorted according to the options specified on the command line	Unused	Used		
-m	Merge sorted input	Unused	Used		
-u	Suppress all but one of the matching keys	Unused	Used		

Factor	Meaning	Levels			
-o output	Output sent to a file	Unused	Valid file	Invalid file	
-T directory	Temporary directory for sorting	Unused	Exists	Does not exist	
-y kmem	Use kmem kilobytes of memory for sorting	Unused	Valid kmem	Invalid kmem	
-z recsize	Specifies record size to hold each line from the input file	Unused	Zero size	Large size	
-dfiMnr	Perform dictionary sort	Unused	fi	Mnr	fiMnr

Factor	Meaning	Levels			
-f	Ignore case	Unused	Used		
-I	Ignore non-ASCII characters	Unused	Used		
-M	Fields are compared as months	Unused	Used		
-n	Sort input numerically	Unused	Used		
-r	Reverse the output order	Unused	Used		
-b	Ignore leading blanks when using +pos1 and –pos2	Unused	Used		

Factor	Meaning	Levels			
-tc	Use character c as field separator	Unused	c1	c1c2	
-k keydef	Restricted sort key definition	Unused	start	end	start type
+pos1	Start position in the input line for comparing fields	Unused	f.c	f	0.c
-pos2	End position in the input line for comparing fields	Unused	f.c	f	0.c
file	File to be sorted	Not specified	Exists	Does not exist	

Fault Model

Create test procedures that will reveal certain kinds of interaction faults between test inputs and test configurations.

An interaction fault is triggered when some combination of t ≥ 1 input values causes the program to enter an invalid state.

The invalid state must propagate to a point where it is observable (reveals the fault).

t = 1 → simple faults
t = 2 → pairwise interaction faults
t-way interaction fault → t-factor fault

Latin Squares

Latin squares and mutually orthogonal Latin squares can be used to select subsets of factor combinations from the full set.

Latin square of order n is an n x n matrix such that no symbol appears more than once in a row and column.

Mathur, Example 6.11. Given S = [A, B], two Latin squares of order 2 are

A	**B**	**B**	**A**
B	**A**	**A**	**B**

Mutually Orthogonal Latin Squares

Two Latin squares are considered mutually orthogonal if the matrix obtained by juxtaposing the two has unique elements.

Abbreviated MOLS.

Mathur, Example 6.13

$$M_1 = \begin{matrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{matrix} \qquad M_2 = \begin{matrix} 2 & 3 & 1 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{matrix}$$

$$L = \begin{matrix} 1\,2 & 2\,3 & 3\,1 \\ 2\,1 & 3\,2 & 1\,3 \\ 3\,3 & 1\,1 & 2\,2 \end{matrix}$$

PDMOLS and IPO

MOLS can be used to construct test factors that test pair-wise combinations of all factors.

- **PDMOLS**

In-Parameter-Order (IPO) procedure can be used to generate mixed-level covering arrays for pair-wise designs.

On-line tools exist to generate orthogonal arrays.

Summary – Things to Remember

Environmental factors can affect program outcomes.

- **test configurations factor in the environmental factors**

Latin squares and mutually orthogonal Latin squares are used to manage the number of test configurations to consider.

Questions and Answers