5. Generation from Combinatorial Design

Andrea Polini

Software Engineering II – Software Testing
MSc in Computer Science
University of Camerino

January 15^{th}, 2015

Combinatorial Design

- Configuration space: all possible settings of the environment variable under which P could be used
- Input space: all possible values that can be taken by input variables

Combination of hardwares, OSs, platforms etc. is generally referred to as compatibility testing

Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

Combinatorial Design

- Configuration space: all possible settings of the environment variable under which P could be used
- Input space: all possible values that can be taken by input variables

Combination of hardwares, OSs, platforms etc. is generally referred to as compatibility testing

Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

Combinatorial Design

- Configuration space: all possible settings of the environment variable under which P could be used
- Input space: all possible values that can be taken by input variables

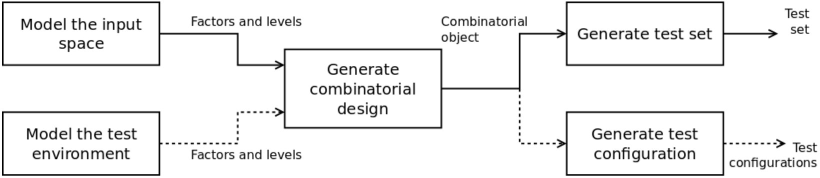
Combination of hardwares, OSs, platforms etc. is generally referred to as compatibility testing

Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

Combinatorial test-design process



Each factor combination may lead to one or more test cases where each test case consists of values of input variables and the expected output. Nevertheless, as usual the generation of all combinations is generally not feasible

Fault model

The approach we are going to discuss targets interaction faults

- interaction faults are triggered when a certain combination of $t \geq 1$ parameter values causes the program containing the fault to enter an invalid state
- faults triggered by some value of input variables regardless of the values of other inputs variables are known as simple faults. When $t = 2$ they are known as pairwise interaction faults. For arbitrary value of t we refer to t-way interaction faults.

Example - 1

Imagine a program that should return the value calculated by different combinations of a couple of functions. In particular when `x=x2` and `y=y2` the returned value should be `f(x,y,z)+g(x,y)`. Now consider the program:

```
begin
  int x,y,z;
  input(x,y,z);
  if (x==x1 and y==y2)
    output(f(x,y,z));
  else
    if (x==x2 and y==y1)
      output(g(x,y));
    else
      output(f(x,y,z)+g(x,y));
end
```

Example - 2

Let $x, y \in \{-1, 0, 1\}$ and $z \in \{0, 1\}$. Are there interaction faults that can be discovered in the following code snippet?

```
begin
  int x,y,z,p;
  input(x,y,z);
  p = (x+y)*z; // instead should be (x-y)*z
  if (p >= 0)
    output(f(x,y.z));
  else
    output(g(x,y));
end
```

Fault vectors and Latin squares

- A fault vector is a k-uple of values for the factors of a program able to trigger a fault. The vector is considered a *t*-fault vector if any $t \leq k$ elements in V are needed to trigger the fault in P.
- A Latin Square of order n is an $n \times n$ matrix such that no symbol appears more than once in a row and a column where the alphabet set Σ as cardinality n.
  e.g. $\Sigma = \{A, B\}$ and $\Sigma = \{1, 2, 3\}$

Latin squares properties

Given a Latin square described by matrix $\mathscr{M}$ a large number of same order matrices can be obtained through row and column interchange and symbol-renaming operations.
A latin square obtained by the mentioned operations is said to be isomorphic to the starting latin square

A latin square can be easily derived using module arithmetic

Latin squares properties

Given a Latin square described by matrix $\mathscr{M}$ a large number of same order matrices can be obtained through row and column interchange and symbol-renaming operations.
A latin square obtained by the mentioned operations is said to be isomorphic to the starting latin square

A latin square can be easily derived using module arithmetic

Mutually orthogonal latin squares (MOLS)

MOLS are a useful tool to generate $t - wise$ vectors from latin squares. Two latin squares are mutually orthogonal if their combination in a matrix of the same order does not generate duplicates.

$MOLS(n)$ indicates a set of MOLS of order n. If n is prime MOLS(n) contains $n - 1$ MOLS and it is referred as a complete set. MOLS exists for each $n > 2 \wedge n \neq 6$

Let's build the MOLS(5) set

Pairwise design - binary factors

Now le's consider tree factors X, Y, Z each one with two levels, and let's generate a pairwise design.

Generalizing the problem on *n* factors each one having two levels.

- we need to define $\mathscr{S}_{2k-1}$ to be the set of strings of lenght $2k-1$ such that each string has exactly k 1s. e.g. $k=3$

	1	2	3	4	5
1	0	0	1	1	1
2	0	1	1	1	0
3	1	1	1	0	0
4	1	0	1	1	1
5	0	1	1	0	1
6	1	1	0	1	0
7	1	0	1	0	1
8	0	1	0	1	1
9	1	1	0	0	1
10	1	0	0	1	1

The SAMNA procedure

Input: n - number of two-valued input variables (factors) Output: A set of factor combinations such that all pairs of input values are covered

1. Compute the smallest integer k such that $n \leq |\mathscr{S}_{2k-1}|$
2. Select any subset of n strings from $\mathscr{S}_{2k-1}$. Arrange these to form an $n \times (2k-1)$ matrix with one string in each row, while the columns contain different bits each string
3. Append a columns of 0s to the end of each string selected
4. Each one of the $2k$ columns contain a bit pattern from which we generate a combination is of the kind $(X_1^*, X_2^*, \ldots, X_n^*)$ where the value of each variable is selected depending on whether the bit in column i, $i \leq i \leq n$ is a 0 or a 1

Example

Consider a simple Java applet named `ChemFun` that allows a user to create an in-memory database of chemical elements and search for an element.

Factor	Name	Levels	Comments
1	Operation	{Create,Show}	Two buttons
2	Name	{Empty,Nonempty}	Data Field, String
3	Symbol	{Empty,Nonempty}	Data Field, String
4	Atomic Number	{Invalid, Valid}	Data Field, data > 0
5	Properties	{Empty,Nonempty}	Data Field, String

Testing for all combinations would require a total of 2^5 tests, but if we are interested for testing for pairwise interactions we can reduce the number of tests to 6.