

# **Functional Testing**

**Neelam Gupta**

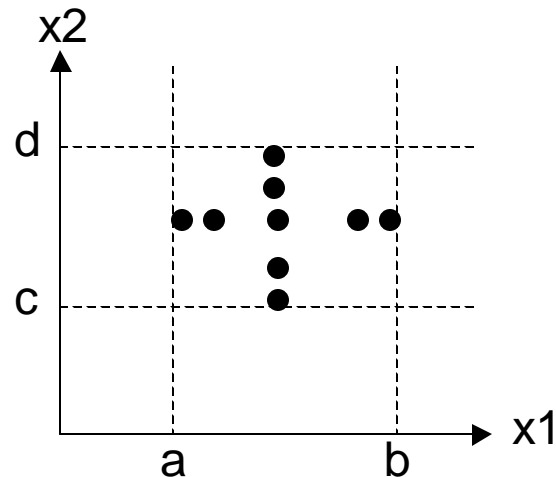
**The University of Arizona  
Tucson, Arizona, USA**

# Boundary Value Testing

Given  $F(x_1, x_2)$  with constraints  $a \leq x_1 \leq b$   
 $c \leq x_2 \leq d$

Boundary Value analysis focusses on the boundary of the input space to identify test cases.

Use input variable value at min, just above min, a nominal value, just above max, and at max.



*Assumption:* Failures are only rarely the result of simultaneous occurrences of two or more faults.

$$\{ \langle X_{1nom}, X_{2min} \rangle, \langle X_{1nom}, X_{2min+} \rangle, \langle X_{1nom}, X_{2nom} \rangle, \\ \langle X_{1nom}, X_{2max-} \rangle, \langle X_{1nom}, X_{2max} \rangle, \langle X_{1min}, X_{2nom} \rangle, \\ \langle X_{1min+}, X_{2nom} \rangle, \langle X_{1max-}, X_{2nom} \rangle, \langle X_{1max}, X_{2nom} \rangle \}$$

*Limitation:* dependence between variables.

# Robustness Testing

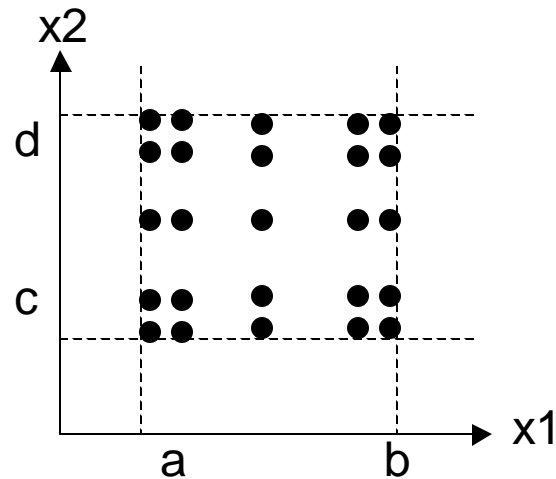
Max+ and Min- also considered in addition to the values already considered for boundary value testing.

# Worst Case Testing

Considers the situation where more than one variable has extreme values.

For each variable make a 5 element set  
{min,min+,nom,max-,max}.

Take Cartesian product of these sets to generate test cases.



# Robust Worst Case Testing

Considers Cartesian product of 7 element set of values for each variable

{min-,min,min+,nom,max-,max,max+}.

## Examples

We will consider two examples to illustrate functional testing techniques:

1. Triangle Problem
2. The Nextdate Program

# Triangle Problem

The triangle program accepts three integers (a,b,c) as input; these are taken to be sides of a triangle. The output of the program is the type of triangle determined by the three sides: Equilateral, Isosceles, Scalene, or not a triangle.

*Scalene* – all sides are of different size

*Equilateral* – all sides equal

*Isosceles* – exactly 2 sides equal

*Not a triangle* – length of one side is greater than or equal to sum of lengths of other two sides.

# The Nextdate Program

It is a function that returns the date of the day after the input date. The month, day and year values in the input date have numerical values with the following constraints.

$$1 \leq \textit{month} \leq 12$$

$$1 \leq \textit{day} \leq 31$$

$$1812 \leq \textit{year} \leq 2012$$

*Note:* A year is a leap year if it is divisible by 4, unless it is a century year. Century years are leap years only if they are multiples of 400. So 2000 is a leap year while the year 1900 is not a leap year.

Case	a	b	c	Exp. Output
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	100	100	Equilateral
9	100	100	100	Isosceles
10	100	199	100	Not a triangle
11	1	200	100	Isosceles
12	2	100	100	Isosceles
13	100	100	100	Equilateral
14	199	100	100	Isosceles
15	200	100	100	Not a triangle

# Equivalence Class Testing

- Complete testing
- Avoiding redundancy

*Equivalence classes* form a partition of a set.

*Partition:* collection of mutually disjoint subsets whose union is the entire set.

*Equivalence testing:* use one element from each equivalence class.

*Key:* choice of equivalence relation.

Program:  $f(a,b,c)$  with input domains A, B, and C.

$$A = A1 \cup A2 \cup A3$$

$$B = B1 \cup B2 \cup B3 \cup B4$$

$$C = C1 \cup C2$$

Elements of partition denoted as:

$a1 \hat{\mathbf{I}} A1$   
 $b3 \hat{\mathbf{I}} B3$   
 $c2 \hat{\mathbf{I}} C2$

# Weak Equivalence Class Testing

- Use one variable from each equivalence class in a test case.

Test Case	a	b	c
1	a1	b1	c1
2	a2	b2	c2
3	a3	b3	c1
4	a1	b4	c2

#test cases = #classes in the partition with the largest numbering of subsets.

# Strong Equivalence Class Testing

- Based on Cartesian product of the partition subsets.

$A \times B \times C$  will have  $3 \times 4 \times 2 = 24$  elements  
 $(a_1, b_1, c_1), (a_1, b_1, c_2), (a_1, b_2, c_1), \dots$

We cover all the equivalence classes and we have one of each possible combination of inputs.

Generalization: equivalence classes on outputs

# Traditional Equivalence Class Testing

Defines equivalence classes in terms of validity.

e.g., valid ranges for next date problem

$1 \leq \text{month} \leq 12; \quad 1 \leq \text{day} \leq 31;$

$1812 \leq \text{year} \leq 2012$

*invalid ranges*

$\text{day} > 31; \text{day} < 1; \text{month} < 1; \text{month} > 12;$

$\text{year} > 2012; \text{year} < 1812$

Given the valid and invalid sets of inputs, the traditional equivalence testing strategy identifies test cases as follows:

- For valid inputs, use one value from each valid class.
- For invalid inputs, a test case will have one invalid value and the remaining values will all be valid.

# Traditional Equivalence Class Test Cases for Next Date Function

Test Case	Month	Day	Year	Expected Output
1	6	15	1912	6/16/1912
2	-1	15	1912	Invalid Input
3	13	15	1912	Invalid Input
4	6	-1	1912	Invalid Input
5	6	32	1912	Invalid Input
6	6	15	1811	Invalid Input
7	6	15	2013	Invalid Input

Equivalence relation defines the class of elements that should be treated in the same way.

Deficiency of traditional approach: same treatment at valid/invalid level.

Better Equivalence relation?

Look at the functionality of the program, that is, what must be done to input date?

Postulate the following equivalence classes:

M1 = {month: month has 30 days}

M2 = {month: month has 31 days}

M3 = {month: month is February}

D1 = {day:  $1 \leq \text{day} \leq 28$ }

D2 = {day: day=29}

D3 = {day: day=30}

D4 = {day: day=31}

Y1 = {year: year = 1900}

Y2 = {year:  $1812 \leq \text{year} \leq 2012$  AND (year  $\neq$  1900)  
AND(year=0 mod 4)}

Y3 = {year: ( $1812 \leq \text{year} \leq 2012$  AND year  $\neq$  0 mod 4)}

# Weak Equivalence Class Test Cases

Test Case	Month	Day	Year	Expected Output
1	6	14	1900	6/15/1900
2	7	29	1912	7/30/1912
3	2	30	1913	Invalid Input
4	6	31	1900	Invalid Input

## Strong Equivalence Class Test Cases

$(m_1, m_2, m_3) \times (d_1, d_2, d_3, d_4) \times (y_1, y_2, y_3)$

$3 \times 5 \times 3 = 45$  test cases

# Equivalence Class Test Cases for the Triangle Problem

Equivalence Class Based on Outputs

R1 = {<a,b,c>: Triangle is equilateral}

R2 = {<a,b,c>: Triangle is isosceles}

R3 = {<a,b,c>: Triangle is scalene}

R4 = {<a,b,c>: a,b,c do not form a triangle}

Case	a	b	c	Exp. Output
OE1	5	5	5	Equilateral
OE2	2	2	3	Isosceles
OE3	3	4	5	Scalene
OE4	4	1	2	Not a triangle

If we define equivalence classes on the input domain, we obtain a richer set of test cases:

$$D1 = \{\langle a,b,c \rangle : a=b=c\}$$

$$D2 = \{\langle a,b,c \rangle : a=b, a \neq c\}$$

$$D3 = \{\langle a,b,c \rangle : a=c, a \neq b\}$$

$$D4 = \{\langle a,b,c \rangle : b=c, a \neq b\}$$

$$D5 = \{\langle a,b,c \rangle : a \neq b, a \neq c, b \neq c\}$$

$$D6 = \{\langle a,b,c \rangle : a \geq b+c\}$$

$$D7 = \{\langle a,b,c \rangle : b \geq a+c\}$$

$$D8 = \{\langle a,b,c \rangle : c \geq a+b\}$$

We could further separate D6 into

$$D6^1 = \{\langle a,b,c \rangle : a=b+c\}$$

$$D6^2 = \{\langle a,b,c \rangle : a > b+c\}$$

Similarly D7 and D8 could be further split into more detailed equivalence classes.