

Functional Testing

Chapter 6

Equivalence Class Testing (ECT)

Equivalence Class Testing

- The use of Equivalence classes as the basis for functional testing has two motivations:
 1. To have a *sense* of “complete” testing
 2. Avoid redundancy (not having serious gaps)
 - Are any of those realized by Boundary Value Testing? Why?
- Equivalence classes form a partition of mutually disjoint subsets (of valid & non valid cases) whose union is the entire set
- Elements in an equivalence class are expected to cause the program to act the same, vs. elements in different classes, we would expect the program to react differently while operating on inputs from different classes.
- The idea of equivalence class testing is to identify test cases by using one element from each equivalence class
- The fact that the entire set is represented provides a form of completeness
- The disjointedness assures a form of non-redundancy
- If the Eq. classes are chosen wisely this greatly reduces redundancy among test cases, why?

Equivalence Class Testing

- The key of equivalence class testing is the choice of the equivalence relation that determines the classes
- Equivalence (Eq.) classes for the triangle problem (on the output range)
 - Equilateral triangle class
 - Isosceles triangle class
 - Scalene triangle class
 - Not a triangle
- For each of the classes one representative is chosen as a test case, we would expect that elements in the same equivalence class cause the program to react similarly
- If we test (5,5,5) and the program works correctly, we would probably not find any problems with (6,6,6)
- If the Eq. Classes are chosen wisely, this greatly reduces the potential redundancy among test cases.
- For a function of 2 variables x_1 and x_2 , where:
 - $a \leq x_1 \leq d$ with intervals $[a,b), [b,c), [c,d]$
 - $e \leq x_2 \leq g$ with intervals $[e,f), [f,g]$
 - Invalid values: $x_1 < a$ & $x_1 > d$ & $x_2 < e$ & $x_2 > g$

Equivalence Class Testing

for the NextDate problem

- For the NextDate problem (equivalence classes are formed on the input domain not the output range as in the triangle problem)
 - Variables:
 - Months (M)
 - Years (Y)
 - Days (D)
 - Equivalence classes for Months $M = M_1 \cup M_2 \cup M_3$
 - 30 day months (M1)
 - 31 day months (M2)
 - February (M3)
 - Equivalence classes for Years $Y = Y_1 \cup Y_2 \cup Y_3 \cup Y_4$
 - Non-century Common years (Y_1)
 - Non-century Leap years (Y_2)
 - Common century years (Y_3)
 - Leap century years (Y_4)
 - Equivalence classes for Days $D = D_1 \cup D_2 \cup \dots?$
 - $D_1 = 1..?$
 - $D_2 = ?$
 -
 -

Equivalence Class Testing for the NextDate problem (cont'd)

- Elements of the partition are denoted:
 - $a_1 \in A_1$
 - $b_3 \in B_3$
 - $c_2 \in C_2$

- m_1 could be any 30 day month (4,6,9 or 11)
- y_3 could be any common century year (1900)
- $d_2 = 28$

- Steps for determining the equivalence classes
 1. Determine the variables that should be tested
 2. Determine the disjoint equivalence classes for each variable according to the difference that should be noticed in program behavior due to each
 3. Disjoint equivalence classes could be done for some or all variables combined if this makes more sense to a specific problem (triangle)

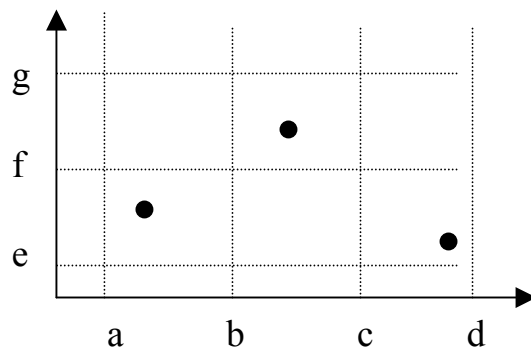
Types of Equivalence Class Testing

- There are four types of equivalence class testing
 1. Weak Normal equivalence class testing
 2. Strong Normal equivalence class testing
 3. Weak Robust equivalence class testing
 4. Strong Robust equivalence class testing

- Normal ECT : only valid entries are considered while determining the test cases
- We consider invalid values in Robust ECT.
- Weak ECT adopts the single fault assumption
- Strong adopts the multiple fault assumption

Weak Normal ECT cont'd

- Weak Normal equivalence class testing:
 - Is accomplished by using one variable from each equivalence class in a test case
 - It ECT follows the single fault assumption



- The number of weak equivalence class test cases is the same as the number of classes in the variable with the largest number of subsets

Test Case	a	b	c
WN1	a ₁	b ₁	c ₁
WN2	a ₂	b ₂	c ₂
WN3	a ₃	b ₃	c ₃
WN4	a ₁	b ₄	c ₄
WN5	a ₂	b ₁	c ₅

Strong Normal Equivalence Class Testing

- Strong Normal Equivalence Class Testing:
 - is based on the multiple fault assumption: the Cartesian product of the partition subsets i.e. the equivalence classes of each variable.
 - In the NextDate problem $M \times Y \times D = 3 \times 4 \times 5 = 60$ test case

Test Case	D	Y	M
SNE1	d_1	y_1	m_1
SNE2	d_1	y_1	m_2
SNE3	d_1	y_1	m_3
SNE4	d_1	y_2	m_1
SNE5	d_1	y_2	m_2
SNE6	d_1	y_2	m_3
SNE7	d_1	y_3	m_1
SNE8	d_1	y_3	m_2
SNE9	d_1	y_3	m_3
SNE10	d_1	y_4	m_1
SNE11	d_1	y_4	m_2
SNE12	d_1	y_4	m_3

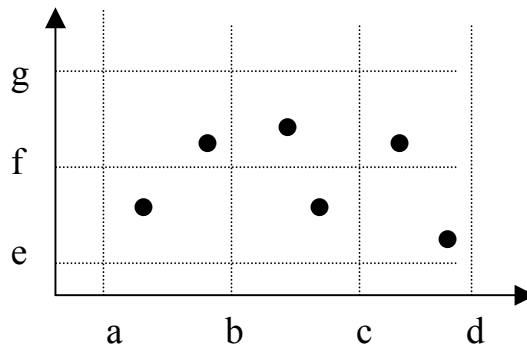
- Test cases SE13-24 are the same but $D = d_2$ instead of d_1 .
 SE25-36 $D = d_3$
 SE37-48 $D = d_4$,
 SE49-60 $D = d_5$

Minimizing the number of Strong Equivalence Class Test Cases

- We could minimize the number of test cases without losing coverage if we merge Y_1 with Y_3 into one equivalence class of common years defined as
 $((\text{year mod } 4 \neq 0) \text{ AND } (\text{year mod } 100 \neq 0)) \text{ OR } ((\text{year mod } 400 \neq 0) \text{ AND } (\text{year mod } 100 = 0))$
And choose 1900 as one of the test cases
- This will reduce the number of test cases to $3 * 3 * 5 = 45$ test cases instead of 60
- The same can be done for leap years, so we end up with 2 equivalence classes for year instead of 4 = $3 * 2 * 5 = 30$ test case instead of 60, a 50% reduction!

Strong Equivalence Class Testing (cont'd)

- Follow the same method in determining strong equivalence class test cases as constructing a truth table in propositional logic
- The Cartesian product covers all the equivalence classes and forms all possible combinations of the inputs
 - This gives a notion of ‘Completeness’:
 - We cover all the equivalence classes and
 - We have a test case of each possible combination of i/ps
- For our 2 variable example:



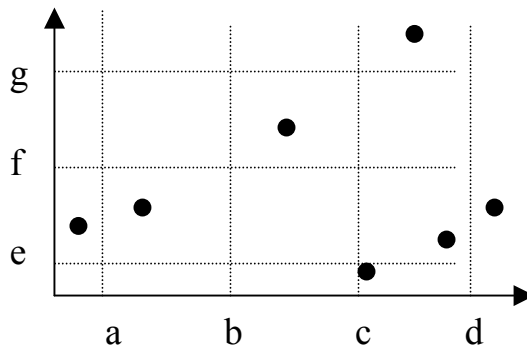
- If exception handling is of high priority, we could extend strong equivalence class testing to include invalid classes: Robust ECT

Weak Robust Equivalence Class Testing

- What does WEAK indicate?
- What does ROBUST indicate?
- In Weak-Robust ECT, we define equivalence classes in terms of validity, i.e. there are two categories:
 - The class of valid inputs
 - The class of invalid inputs
- So far in our examples we have partitioned only the valid inputs into equivalence classes
- For example in the NextDate function, month:
 - Valid month = 1..12
 - Invalid months are
 - An integer < 1 or > 12
 - or any non-integer
 - A real number
 - An alpha-character
 - A special character
 - Null, etc.

Weak Robust Equivalence Class Testing (cont'd)

- Given these valid and invalid sets of inputs, the traditional equivalence testing strategy identifies test cases as follows:
 - For valid inputs of all variables, do weak equivalence class testing (all vars are valid)
 - For invalid inputs, assume only one of the variables to be invalid, and the others valid (why?)

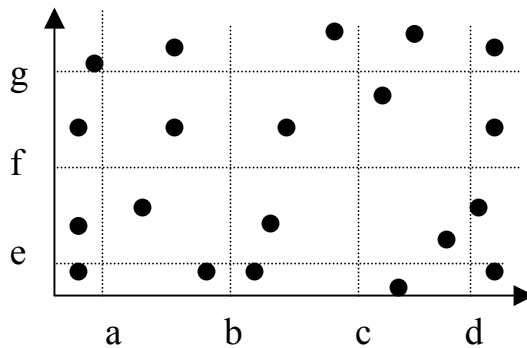


- The only time it makes sense to use the weak robust approach is when the implementation language is not strongly typed

Strong Robust Equivalence Class Testing

- Strong?
- Robust?

- We obtain test cases from each element of the Cartesian product of all the equivalence classes



Examples of Equivalence Classes

The Triangle Problem

- We have already showed the equivalence classes for the triangle problem on the output range, generating only four test cases

Test Cases	a	b	c	Expected o/p
WN1				
WN2				
WN3				
WN4				

- There are no valid sub-intervals of variables a,b, & c. Hence Strong Normal EC test cases are identical to the weak normal EC test cases...How?
- Develop Weak Robust Equivalence class test cases for the triangle problem. How many test cases should be generated?

Examples of Equivalence classes

The Triangle Problem cont'd

- What do we need to add to the cases we generated for Weak robust ECT to generate the test cases for strong robust ECT

Equivalence Classes for the Triangle Problem (input domain)

- Now we would like to see equivalence classes for the triangle problem on the input domain
- Instead of separating the variables and doing equivalence classes for each variable alone resulting in classes of valid & invalid inputs only, we could combine these inputs:

– $EC1 = \{ \langle a, b, c \rangle : a = b = c \}$ ← **Equilateral triangle**

– $EC2 = \{ \langle a, b, c \rangle : a = b, a \neq c \}$ ▼

– $EC3 = \{ \langle a, b, c \rangle : a = c, a \neq b \}$ ← **Isosceles triangle**

– $EC4 = \{ \langle a, b, c \rangle : b = c, a \neq b \}$ ←

– $EC5 = \{ \langle a, b, c \rangle : a \neq b, a \neq c, b \neq c \}$ ← **Scalene triangle**

– $EC6 = \{ \langle a, b, c \rangle : a \geq b + c \}$ ▼

– $EC7 = \{ \langle a, b, c \rangle : b \geq a + c \}$ ←

– $EC8 = \{ \langle a, b, c \rangle : c \geq a + b \}$ ▼

Not a triangle

– What if $a = 1$, $b = 1$ and $c = 5$

- i.e. $a = b$, $a \neq c$ **and** $c \geq a + b$
- Is this an equilateral triangle or not a triangle?

Examples of Equivalence Class Test Cases

Next Date Problem

- For the Next Date Problem, there are 5 valid day equivalence classes and 3 valid month equivalence classes and 4 valid year equivalence classes.
- How many Weak Normal EC test cases will be generated? Give examples.
- How many Strong Normal EC test cases will be generated? Give examples

Combining Boundary Value Analysis with Equivalence Class Testing

- Equivalence class testing is strengthened by a hybrid approach with boundary value testing
- For each equivalence class select the elements representing the class as you would in boundary value analysis, select the min, min+, nom, max- & max.
- Robust testing could be used to show the effects of invalid test cases when using min- and max+ like in the month variable in the NextDate problem