Software Development II
Unit 4: Blackbox Testing
Equivalence Class Testing

Markus Roggenbach

February 2020

You will learn

An advance blackboard testing technique

In particular, we look into

- how to partition the input domain
- how to choose representatives of each partition in order to obtain a test suite

Equivalence class testing

Boundary Value Testing ignores the output

Given a computational problem with 2 inputs of type `byte`, the testsuite will always exercise the SUT for the same inputs, namely

name	x	y	expected result
T1	-128	-4	
T2	-127	-4	
T3	-4	- 4	
T4	126	-4	
T5	127	-4	
T6	-4	-128	
T7	-4	-127	
T8	-4	126	
T9	-4	127	

independent of what we respect as result:

$x * y, x + y, x - y, x^y, \ldots$

In contrast: analysis triggered by the output

e.g.: tests should mirror elementary algebraic properties

Examples:

Multiplication:

- 1 * x = x
- 0 * x = 0
- a * b > 0 (a,b > 0 or a,b < 0)
- a * b < 0 (one of a,b < 0 and one of a,b > 0)

Square function:

- $x^2 > 0 \ (x > 0)$
- $x^2 > 0 \ (x < 0)$
- $0^2 = 0$
- $1^2 = 1$

Rationale for equivalence class testing

Assumption:
The SUT will behave
“the same”
for certain classes of inputs.

If we cover all possible behaviours, the system is well tested.

Def: Equivalence relation

A relation $\sim \subseteq X * X$ on a set X is called an equivalence relation, if $\sim$ is

- reflexive,
- symmetric, and
- is transitive.

Def: Partition

A set P of nonempty sets is a partition of a set X if

1. The union of the elements of P is equal to X.
   (We say the elements of P cover X.)
2. The intersection of any two distinct elements of P is empty. (We say the elements of P are pairwise disjoint.)

Equivalence relation induced by a partition

Let P be a partition of a set X.

Definition:
Let $x, y \in X$.
Define a relation $\sim \subseteq X \times X$ by
$x \sim y$ iff there exists $R \in P$ with $x, y \in R$.

Theorem: $\sim$ is an equivalence relation.

Partion induced by equivalence relation

Let $\sim \subseteq X * X$ be an equivalence relation on a set X.

Definition:
For $x \in X$ define $\bar{x} := \{y \in X \,|\, x \sim y\}$.

Theorem: $P := \{\bar{x} \,|\, x \in X\}$ is a partition of X.

Basic idea: weak normal equivalence class testing

Step 1 Define partitions for either the whole input domain or each variable

Step 2 Identify test cases by randomly choosing one element from each equivalence class – in case the whole input domain was partitioned: these are the test inputs – in case the domain of each variable was partitioned: combine these elements in such a way that for each class there is at least one test case.

Triangle Problem – an example where the whole input domain is partioned

$$
\begin{aligned}
R1 &= \{(a,b,c) \mid \text{triangle with sides } a,\ b, \text{ and } c \text{ is equilateral.}\}\\
R2 &= \{(a,b,c) \mid \text{triangle with sides } a,\ b, \text{ and } c \text{ is isosceles.}\}\\
R3 &= \{(a,b,c) \mid \text{triangle with sides } a,\ b, \text{ and } c \text{ is scalene.}\}\\
R4 &= \{(a,b,c) \mid \text{sides } a,\ b, \text{ and } c \text{ do not form a triangle.}\}
\end{aligned}
$$

Test Case	a	b	c	expected output
WN1	5	5	5	equilateral
WN2	2	2	3	isosceles
WN3	3	4	5	scalene
WN4	4	1	53	not a triangle

Speed control (old exam question) – Computational Problem

SpeedControl:

Input: integers $0 \leq l \leq 70$ and $0 \leq s \leq 200$

Output: “under speed limit”, if $s \leq l$

“marginally over speed limit “, if $l < s \leq 1.05 * l$

“significantly over speed limit “, otherwise

1. Define equivalence classes for the whole input domain according to the outcome of the computational problem **SpeedControl**.
2. Write a minimal test suite for Equivalence Class Testing.

Speed control: Equivalence Classes

- $C_1 = \{(l, s) \in \mathbf{Z} \times \mathbf{Z} \,|\, 0 \leq l \leq 70,\ 0 \leq s \leq 200,\ s \leq l\}$
- $C_2 = \{(l, s) \in \mathbf{Z} \times \mathbf{Z} \,|\, 0 \leq l \leq 70,\ 0 \leq s \leq 200,\ l < s \leq 1.05 * l\}$
- $C_3 = \{(l, s) \in \mathbf{Z} \times \mathbf{Z} \,|\, 0 \leq l \leq 70,\ 0 \leq s \leq 200,\ 1.05 * l < s\}$

Speed control: Test Suite

Name	l	s	expected output
T1	10	5	under speed limit
T2	50	52	marginally over speed limit
T3	50	60	significantly over speed limit

Keyboard driver (old exam question) – Computational Problem

For a keyboard driver it is necessary to classify keystrokes into "lowerCaseCharacter" (i.e. the characters "a", "b", . . . , "z"), "digit" (i.e. "0", "1", . . . , "9"), "rest".

KeystrokeType:

Input: character c of type `char`

Output: lowerCaseCharacter, if c is one of "a", "b", . . . , "z".
digit, if c is one of "0", "1", . . . , "9".
rest, otherwise

1. Define the equivalence classes for the input domain `char` by partioning the whole input domain according to the outcome of the computational problem **KeystrokeType**.
2. Write a test suite for Equivalence Class Testing.

On the blackboard: Equivalence Classes & Test Suite

How to find the classes?

In the previous three examples always:

- demonstrate all possible outcomes of a classification problem

Later on:

- Problem analysis

Notation for intervals

Over the numbers (reals, rationals, integers, naturals), the following set notations are often used:

- $[a, b] = \{x \mid a \leq x \leq b\}$
- $(a, b) = \{x \mid a < x < b\}$
- $[a, b) = \{x \mid a \leq x < b\}$
- $(a, b] = \{x \mid a < x \leq b\}$

Note: if $b < a$, then $[a, b] = \{\}$.

Illustration of Step 1 on a Computational Problem with two parameters

F:

Input:	x and y
Output:	$\ldots$

Assume that

- $[a, b), [b, c](c, d]$ is a useful partition for x.
- $[e, f), [f, g]$ is a useful partition for y.

with $a \leq b \leq c \leq d$ and $e \leq f \leq g$.

Illustration of Step 2 on a Computational Problem with two parameters

Then we need 3 test cases, e.g.,

Test Case	x	y	expected output
T1	a	e	...
T2	b	f	...
T3	d	f	...

T1 covers $[a, b)$ and $[e, f)$.
T2 covers $[b, c]$ and $[f, g]$.
T3 covers $(c, d]$ and $[f, g]$.

What you have learned in this unit

Definitions

- Equivalence relation
- Partition
- Process of deriving a equivalence class testsuites as UML Activity Diagram

Theorems

- Every equivalence relation gives rise to a partition.
- Every partition gives rise to an equivalence relation.

You should be able to explain by example

- Why equivalence class testing is useful in ‘classification problems’
- How to design test suites for four variants of equivalence testing