ECE 322
Lab Report 3

Arun Woosaree
XXXXXXX

October 20, 2019

1 Introduction

The purpose of this lab was to serve as a practical introduction to some more black box testing techniques. In this lab, the testing methods introduced were the Extreme Point Combination (EPC), and the Weak n x 1 strategy methods. We tested two programs written in Java using both of these testing strategies. The first application, named Drone takes in three arguments, and outputs either 'Success!', 'Failure', or an error message based on whether the arguments are integers ≥ 0, and their sum is less than $k = 100$. The second program, named RemoteCar takes in two arguments which represent points on a Cartesian plane, and outputs either 'Ok', 'Out of range', or an error message based on whether the point is on a circle of radius 1 about the origin. The idea for EPC testing is to identify the input domain limits, and produce all possible combinations of inputs with each of the input variables taking on a minimum value, slightly under minimum, a maximum value, and slightly over maximum value. One additional test case is added somewhere within the valid subdomain to generate a total of $4^n + 1$ test cases, where n is the dimension of inputs, or put more simply, the number of input variables. With the weak n x 1 strategy, we attempt to find linear boundaries of the problem using domain analysis. n points are selected on each linear boundary, where n is the number of input variables, and one additional point is chosen outside of each boundary. An additional point within the boundary is chosen if the boundary is open, and if the boundary is instead closed an additional point outside the boundary is chosen. In the end, with the weak n x 1 strategy, $b \times (n + 1)$ test cases are generated, where b is the number of linear boundaries, and n is the dimensionality, or the number of inputs.

kms

2 Task 1

For task one in this lab, the InsuranceCalculator application was tested using a cause-effect graph from which a decision table and test cases were made.

InsuranceCalculator is a GUI program written in java, which takes in three inputs:

- gender: Male/Female
- age: integer
- : claims: integer

Depending on the value of the inputs, the program will output either $750, $1000, $1500, $3000, or ERROR.

From the following rules, a cause-effect graph was generated, and can be found in Appendix A.

- IF sex = Male AND age < 25 AND claims $= 0$ THEN premium = $1500
- IF sex = Male AND $25 \leq$ age < 65 AND claims $= 0$ THEN premium = $1000
- IF sex = Female AND age < 65 AND claims $= 0$ THEN premium = $750
- IF age ≥ 65 AND claims $= 0$ THEN premium = $1500
- IF claims ≥ 1 THEN premium = $3000

From the cause-effect graph found in Appendix A, the following decision table was made:

	1	2	3	4	5
Causes					
A	0	1	1	x	x
B	1	0	0	x	x
C	x	0	1	0	x
D	x	1	0	0	x
E	0	0	0	1	x
F	1	1	1	1	0
G	0	0	0	0	1
Effects					
Y1	1	x	x	x	x
Y2	x	1	x	x	x
Y3	x	x	1	1	x
Y4	x	x	x	x	1

From the decision table above, the following test cases were made:

Test ID	Sex	Age	Claims	Expected	Actual
1	F	64	0	$750	$750
2	M	25	0	$1000	$1000
3	M	24	0	$1500	$1500
4	F	66	0	$1500	$1500
5	F	54	1	$3000	$3000

Normally, failed test cases would be highlisghted in red. However, none of the test cases failed in this case.

2.1 Discussion

Using both the EPC and the weak n x 1 strategy, we see that for all of the failed test cases, the input variable x_2 is a negative. For negative inputs, the program is expected to give an error telling the user that negative inputs are not valid. Instead, the program outputs 'Success!' or 'Failure!' when $x_2 < 0$. Using the EPC strategy, we see 2 types of failures, where the program outputs both 'Success!' or 'Failure!'; however, with the weak n x 1 strategy we only saw the case where 'Success!' is outputted mistakenly. It should be noted however, that both testing strategies found an error which seems to stem from the same issue, (the application not checking if x_2 is negative, and the weak n x 1 strategy did so with less test cases (17 versus 65). As a result, for this problem, the weak n x 1 strategy seemed to be both more efficient and effective, since the same problem was caught using the EPC strategy but with far fewer test cases.

kms

3 Task 2

For task two in this lab, the BoilerShutdown application was tested using a simplified cause-effect graph from the one which was given, from which a decision table was made, and test cases were created. BoilerShutdown is a GUI application written in Java, which has 10 checkbox inputs labelled alphabetically from A to J. These are defined as:

- A - Water level meter failure during operation
- B - Steam rate meter failure during operation
- C - Water level out of range
- D - Instrumentation system failure
- E - Communication link failure
- F - Steam rate meter failure during startup
- G - Water level meter failure during startup
- H - Multiple pumps failure (more than one)
- I - Confirmed keystroke entry

- J - Confirmed “shut now” message

The given cause-effect graph for Task 2 was simplified, and can be found in Appendix B. By tracing back the graph, we were able to determine that inputs A, B, and C had no effect on whether the Boiler would shut down or not. Furthermore, We were able to eliminate almost all intermediate nodes, since it was found that the boiler effectively shuts down if any of the inputs D, E, F, G, H, I, or J = 1 and it does not shut down if all of the inputs from D-J = 0. From the new cause-effect graph, the following rules were derived:

- IF D OR E OR F OR G OR H OR I OR J THEN Boiler Shutdown
- IF NOT D AND NOT E AND NOT F AND Not G AND NOT H AND NOT I AND NOT J THEN Boiler not Shutdown

From the rules defined above, a decision table was made:

	1	2	3	4	5	6	7	8
Causes								
A	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x
D	1	x	x	x	x	x	x	0
E	x	1	x	x	x	x	x	0
F	x	x	1	x	x	x	x	0
G	x	x	x	1	x	x	x	0
H	x	x	x	x	1	x	x	0
I	x	x	x	x	x	1	x	0
J	x	x	x	x	x	x	1	0
Effects								
Boiler shutdown	1	1	1	1	1	1	1	0
Boiler not shutdown	0	0	0	0	0	0	0	1

From the decision table above, the following test cases were made:

Testid	A	B	C	D	E	F	G	H	I	J	Expected	Actual
1	0	0	0	1	0	0	0	0	0	0	FAIL	FAIL
2	0	0	1	0	1	0	0	0	0	0	FAIL	FAIL
3	0	1	0	0	0	1	0	0	0	0	FAIL	FAIL
4	0	1	1	0	0	0	1	0	0	0	FAIL	FAIL
5	1	0	0	0	0	0	0	1	0	0	FAIL	FAIL
6	1	0	1	0	0	0	0	0	1	0	FAIL	FAIL
7	1	1	0	0	0	0	0	0	0	1	FAIL	FAIL
8	1	1	1	0	0	0	0	0	0	0	Ok	Ok

Normally, failed test cases would be highlighted in red. However, all test cases passed in this scenario.

3.1 Discussion

. . . however, using this method, some cases were missed. For example,

The domain approximation is somewhat effective. It seems to work best if test inputs chosen are far away from the boundaries, since there is a risk of a false negative test result if the chosen test input is too close to the boundary. This can happen when the test case chosen is outside of the approximated boundary, while it is still in the actual subdomain's boundary. In such a scenario, we might expect a test case to fail using the approximation, yet if the program is running correctly, the test case will pass. The configuration could have been adjusted to yield higher accuracy of the subdomain. For example, we could have increased the number of linear boundaries. Using the EPC strategy, the complexity of testing is unaltered. There are still 17 test cases, and they would all be the same. However, using the weak n x 1 strategy, the complexity would be increased. We would have to make $b \times (2 + 1) + 1$ test cases, where b is the number of linear boundaries we choose to use for the approximation. In other words, we would need an additional 3 test cases for each boundary we add to increase the accuracy.

For this problem, it seems that the EPC testing strategy is more accurate. Using the weak n x 1 strategy, we ran into some cases where the test input was outside of our approximated boundary, yet it was inside the actual subdomain. This resulted in 4 test case failures, which were not actual errors with the program itself, but rather due to the innacurate linear approximation of the subdomain. As a result, no actual errors were identified in the application. In this case, with $m = 4$ boundaries, the number of test cases for the EPC and weak n x 1 strategies were similar. However, the weak n x 1 strategy could have a lot more test cases if we added more linear boundaries to make a better approximation of the problem's subdomain.

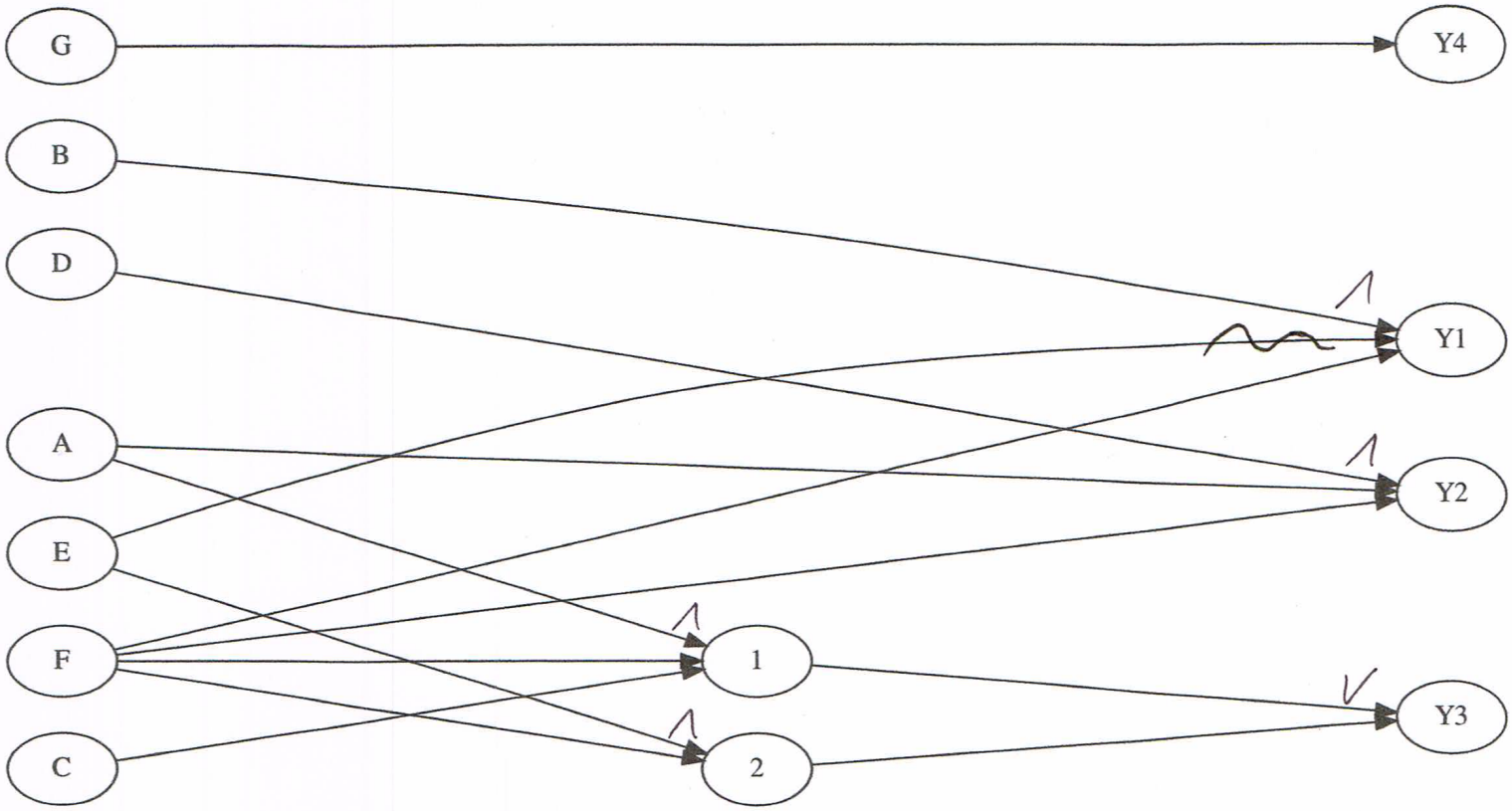
kms

4 Conclusion

In this lab, we were introduced to two more black box testting strategies. The techniques learned were the Extreme Point Combination, or EPC strategy, and the weak n x 1 strategy. It would not be fair to say that one testing strategy is universally better than the other, since it would be better to match each problem at hand to the appropriate testing strategy. The weak n x 1 strategy seems to shine when the problem's subdomain is already linear in nature. This way, no approximation needs to be made, and we get very few test cases $b \times (n + 1) + 1$, where b is the number of boundaries, and n is the number of input variables. The weak n x 1 strategy appears to not be very effective however, when the problem's subdomain is approximated as a collection of linear boundaries. This configuration appears to result in false negative test cases, and additionally, if

more input boundaries are used to increase the accuracy of the input domain, it is still not perfect, and furthermore, $n + 1$ additional test cases need to be added for each additional boundary, which can result in more test cases required to be carried out, which all still have the potential issue of resulting in a false negative result. On the other hand, the EPC strategy might seem tedious at first, generating $4^n + 1$ test cases, which might seem excessive. However, it seems to overall be more accurate than the weak n x 1 stratrgy. There were no false positive test cases, and furthermore, more failure scenarios were found, even though they are likely from the same root cause. (For the Drone program, we found using the EPC strategy 2 types of failures, where the program outputs both 'Success!' or 'Failure!' when x_2 was negative instead of an error message, however, with the weak n x 1 strategy we only saw one case where 'Success!' was outputted mistakenly. From a purely subjective standpoint, I personally prefer the EPC testing strategy over weak n x 1, since even though a lot of test cases are typically generated, this step can be automated using a script, and not lot of thought, since only four values (min, slightly under min, max, slightly over max) plus one additional valid test case needs to be thought of. Computers are fast, so we can simply generate most of the test cases without much thought, and then compare the results with ease using more automation. However, using the weak n x 1 strategy, one has to really think about each boundary, and choose n points on it, plus one point slightly outside each boundary, and additionally, one final test case. Additionally, weak n x 1 testing does not seem to work well for subdomains that are non-linear in nature, while the EPC strategy seems to overall work well and reliably.

Appendix A: Cause Effect Graph for Car Insurance Premium



Appendix B: Simplified Cause Effect Graph for Boiler Shutdown