

Software Testing MTAT.03.159

Lab 3: Combinatorial Testing

Institute of Computer Science, University of Tartu

March 2018

Introduction

Combinatorial testing is a black box testing technique for systematically testing t-way interactions of input or configuration parameter values. This is important, because it has been found that testing only 1-way interactions (each value at least once in isolation) will most likely not discover all or even most of the defects in a system. It is most commonly used to test architecturally complex systems with many configuration parameters, but it is useful for testing any system with a multitude of input parameters. The number of test cases increases exponentially when increasing interaction strength t , but logarithmically with the number of parameters. Combinatorial testing with the help of tools like the ACTS tool allow for a fast generation of t-way interaction test cases that one can use to systematically evaluate the system under test, saving time and minimizing user error.

Part A (Lab session)

1. You are given a toy application (found inside “MarriageChecker.zip”) that is used to check whether 2 people are married or not. We don’t have the full oracle, but we know that the application should return that the 2 people are not married if they are people of the same sex. This can be assumed, because the application uses test data based on a country that doesn’t permit it. Using this information, it is up to you to check whether or not this application has any failures and if so, how many? Before you start, proceed to the next step.
2. What is the minimum number of test cases needed to cover all 1-way interactions (for each input variable, each value is assigned at least once)?
3. Execute test cases necessary to cover 1-way interactions and see if you find any failures.
4. How many test cases are required to test all possible combinations (exhaustive testing)?
How many to cover all 2-way interactions (for all pairs of variables, all possible value combinations are executed at least once)?
How about 3-way interactions?

5. The lab materials also contain a ZIP file called “ACTS” – inside you can find a JAR file for the ACTS tool. Or alternatively, you can download the ACTS tool using this link:

https://csrc.nist.gov/CSRC/media/Projects/Automated-Combinatorial-Testing-for-Software/documents/download/ACTS3_0.zip

The zip file also includes a user guide, should you need any assistance with the tool.

6. Get acquainted with the ACTS tool and see how many test cases are really needed to cover all 2-way interactions.

Follow these steps in order to create a new system and start creating your covering array:

- 6.1. With the ACTS tool opened, click “System” and choose “New”
- 6.2. Enter the System Name (e.g. “Marriage checker”)
- 6.3. Enter the Parameter Name found in the toy application (e.g. “First_first_name”)
- 6.4. Enter the Parameter Type (in the case of this application, Enum)
- 6.5. Enter the Simple Value (e.g. “Brad”)
- 6.6. Press “Add ->”
- 6.7. Repeat steps 6.5 and 6.6 for each value of the parameter
- 6.8. Press “Add to table”
- 6.9. Repeat steps 6.3 to 6.8 for each parameter found in the toy application. You should end up with this:

The screenshot displays the ACTS tool interface. On the left, the 'System' configuration panel is visible, showing the 'System Name' as 'Marriage checker'. Below this, the 'System Parameter' section includes fields for 'Parameter Name' and 'Parameter Type' (set to 'Enum'). The 'Parameter Values' section shows the 'Selected Parameter' as 'Enum', with 'Simple Value' and 'Range Value' (0 to 1) fields. Buttons for 'Add ->', '<- Remove', 'Base Choice', 'Invalid Value', and 'Add to Table' are present. On the right, the 'Saved Parameters' table lists four parameters: 'First_first_name', 'First_last_name', 'Second_first_name', and 'Second_last_name', all of type 'Enum'. The 'Parameter Value' column contains lists of values for each parameter. At the bottom, 'Add System' and 'Cancel' buttons are visible.

Parameter Name	Parameter Type	Parameter Value
First_first_name	Enum	[Brad,Keanu,Audrey,Angelina]
First_last_name	Enum	[Pitt,Reeves,Hepburn,Jolie]
Second_first_name	Enum	[Tyler,John,Holly,Lara]
Second_last_name	Enum	[Durden,Wick,Golightly,Croft]

- 6.10. Once you have added all the necessary parameters and their values, press “Add System”.
 - 6.11. With the CT System created, click “Operations” and choose “Build”
 - 6.12. Now choose the interaction strength of the covering array and the algorithm that is used to create it. For this task choose the IPOG algorithm and strength 2.
 - 6.13. Press “Build”, once you have chosen the algorithm and desired interaction strength
7. Run all test cases generated by the ACTS tool and see if you find any failures. Please record for each executed test case whether you found a failure or not.
 8. When using the toy application, you probably noticed that the parameters which determine the sex of the first and second person is their first name. This implies that there is a higher degree of interaction between those parameters. The ACTS tool also has a feature called “Mixed strength” which allows different parameter groups to be covered with different strengths. This can be useful for reducing the amount of test cases generated when there is suspicion that some parameters interact with each other more than others. Set relations for “Mixed strength” using these steps:
 - 8.1. Ensure that in your “System view”, your system folder is checked (“[SYSTEM-*system name*]”)
 - 8.2. Click “Edit” and choose “Modify”
 - 8.3. Navigate to the “Relations” tab
 - 8.4. You can now select several parameters in order to create a relation with a desired strength (the strength can’t be higher than the amount of parameters selected)
 - 8.5. Create a relation between all parameters with a strength of 1
 - 8.6. Create a relation with a strength of 2 between the 2 parameters which you think have a higher degree of interaction.
 - 8.7. Click “Modify system”
 - 8.8. Now build the system by using “Mixed” as the strength.
 9. Compare this new covering array to the one created in step 7 – how much has the total amount of test cases decreased? Have you lost any test cases that triggered a failure? When do you think this feature would be useful to use?

Part B (Homework)

Introduction & preparation

1. You are given another toy application (found inside “Booking.zip”), the main purpose of which is to display how many listings correspond to a search criteria. Imagine that there is a database connected to the application and it’s being constantly updated – so checking for results multiple times using the same search criteria won’t necessarily display the same result. There are 2 ways of using the application – you can either manually input your search parameters to see how many matching results were found, or you can import a CSV file named “acts-output.csv” containing all of the desired queries and the application will proceed to automatically generate a file named “booking-output.csv” including the input of the initial imported CSV file and the results to all of the queries.

NB! The rules regarding the CSV file that the Booking application can import are described in step 3. Additional rules are defined in the tasks below.

2. Get acquainted with the parameters, possible input values of the application and the output. Some known rules about the application (the Booking application should return “0 results” for queries that violate these rules):
 - 2.1. Only a maximum of 2 people can be booked in a shared room or private room.
 - 2.2. A villa can’t be booked in the city centre.
 - 2.3. All hotels offer service by default.

3. Rules and guidelines for the tasks ahead:

- 3.1. When creating a new Combinatorial Testing System, in order for it to work with the application later when importing, **make sure the parameter names and values match their respective labels and values in the application** (labels are Name, People, Accommodation, Region, Year, Month, Week, Duration and Service) and that **all parameters are represented**. Capitalization doesn’t matter though.
- 3.2. The **Service** parameter should be of Boolean type in the CT system.
- 3.3. You must also **save your CT system as an XML file** – this is required for tasks 1 & 2 and will also be used in the grading process.
- 3.4. Important when building your covering arrays
 - 3.4.1. The booking application doesn’t support algorithms other than IPOG or IPOG-F because “Randomize DontCare Values” must be checked in order for the parameters to always have a specific value. **Make sure you don’t uncheck “Randomize DontCare Values”** when building the covering arrays.
 - 3.4.2. If you want **interaction strength t that is greater than 6**, you have to modify your CT system, navigate to the Relations tab and set strength t for all parameters.
- 3.5. You can export your covering array by selecting Operations -> Export -> CSV format.
 - 3.5.1. **Please name these files properly** (e.g. “acts-output-3-IPOG.csv” for a test set built using the IPOG algorithm with strength 3), because you will be exporting a

number of them, so it will make it easier for you to remember which file you just imported in the Booking application, unless you create and import them one at a time.

- 3.5.2. The Booking application recognizes commas as a separator for parameters and parameter values, so inspect your exported CSV file should you run into any problems with it not importing the acts-output covering array properly.
- 3.6. **Don't delete the CSV file generated by the Booking application** – it will be used in the grading process. Instead **rename them to indicate what strength level and algorithm was used** (e.g. “booking-output-3-IPOG.csv” if the output was generated from a covering array with a strength level of 3 using the IPOG algorithm).

Task 1

Create a new Combinatorial Testing System in the ACTS tool and use it to test the Booking application. Generate covering arrays from strength 1 to strength 9, using both the IPOG and the IPOG-F algorithms. Should you be confused as to what the general workflow looks for this task, see the model below (Appendix 2). Complete the following steps:

1. Save your CT system as a separate XML file named “**booking.xml**” by pressing “System” and choosing “Save As” – this is necessary for the grading process.
2. Generate & export as CSV files the covering arrays using both the IPOG and IPOG-F algorithms, from strength 1 to strength 9. Refer to the rules & guidelines above for **how to set your interaction strength higher than 6**.
 - When exporting, rename them to reflect the strength and algorithm used (e.g. “acts-output-8-IPOG.csv”) to make it easier for you to recognize which file you’re about to use with the Booking application.
3. **Rename the “booking-output.csv” files**, that the Booking application outputs, according to the strength and algorithm used (e.g. “booking-output-8-IPOG.csv”). **Make sure to do this after every import**, otherwise the application will overwrite your previous results.

Analyze the output:

 - Fill out the table in Appendix 1 for IPOG and IPOG-F algorithms separately
 - The values are separated by commas, so you can use Excel’s “Text to Columns” feature to separate it into cells, making it easier to sort through the data.
4. Construct a graph displaying the amount of **cumulative unique** failures up to strength t (x-axis = strength, y-axis = no. of unique failures). Example: if you have found 1 unique failure at strength 1, but the same failure at strength 2, the number of cumulative unique failures up to strength 2 is still 1.

Note: In this task, by “**failure**” we mean an error message auto-generated by the Booking application.

 - Do this for both the IPOG and IPOG-F algorithms. Displaying both results on one graph is better, should you need to analyze them when solving the upcoming tasks.

Task 2

Conducting tests with tens of thousands of test cases, even when automated, is often just not possible. In these cases, one needs to find ways to be more efficient. Do the following:

1. Using **only** the ACTS tool, build a covering array containing at most 250 test cases that finds the highest amount of **unique failures**. You are not allowed to change the exported CSV file. Remember that the booking application doesn't support algorithms other than IPOG or IPOG-F and that "Randomize DontCare Values" must be checked.
Note: In this task, once again, by "**failure**" we mean an error message auto-generated by the Booking application.
 - 1.1. Recall what features the TA introduced to you in the lab that might be relevant here or use the ACTS user guide to see what features you can implement to approach this task.
 - 1.2. Once you have achieved this, save your CT system as a separate XML file named "**booking-mixed.xml**" by pressing "System" and choosing "Save As" – this is necessary for the grading process.
 - 1.3. **Don't delete your "booking-output.csv" file** – it will also be used in the grading process. **Instead rename it "booking-output-mixed.csv"**.
2. Explain the reasoning behind the CT system you built to achieve the objective of step 1.

Task 3

As can be the case with real applications, the description offered for our toy application does not reveal a specific test oracle that we can use to verify whether the output produced by the application is actually correct. However, as a proxy, we can try to check whether any of the business rules offered in step 2 have been violated.

Analyze the largest covering array (strength 9) used in task 1 and answer these questions:

1. Do you think the Booking application violates the business rules introduced in step 2?
2. If so, which rule(s) does it violate and how many times?
3. How did you determine this amount?
4. Provide a file containing all of the rule violations as a separate file, named "**booking-violations.csv**".

An example of a rule violation is the following: more than 2 people book a private or shared room and the application returns a result other than 0.

Note 1: In this task, we are not interested in analyzing failures that correspond to auto-generated error messages from the Booking application. In this task, you have to think about how to find incorrect outputs of the application without having a defined test oracle.

Note 2: It's not possible to do this task purely manually, because there are too many test outputs to check. Therefore, try to think of an automated or semi-automated approach for checking rule violations and use it to complete this task.

Reporting

Submit a report consisting of:

- Task 1:
 - Statistics demonstrating the amount of unique failures found up to strength t in the form of a graph(s) for each algorithm used - as separate graphs or as one.
 - The filled tables for both algorithms (Appendix 1).
 - Generated “booking-output.csv” files **which are renamed to reflect the strength and algorithm used.**
 - Your CT system file named “booking.xml”
- Task 2:
 - Your answers to questions found in Task 2
 - The necessary “booking-mixed.xml” and “booking-output-mixed.csv” files
- Task 3:
 - Your answers to the questions
 - A CSV file containing all rule violations to back up your argument

NB! Make sure you present the necessary CSV and XML files, otherwise you will get 0 points for the respective tasks!

Grading

You can get up to 10 points for this lab, plus 1 bonus point from task 2.

The grading is as follows:

1. 1 point for being active in the lab
2. Up to 4 points for Task 1
 - 2.1. Up to 1.5 point for the IPOG table and up to 1.5 point for the IPOG-F table
 - 2.2. Up to 1 point for the graph
 - Up to 0.5 for each algorithm
 - 2.3. **No matter whether correct tables and graphs were submitted, you get 0 points if the relevant CSV files are not submitted, too**
3. Up to 3 points for Task 2 plus 1 bonus point.
 - 3.1. Up to 3 points for finding at least 2 unique failures (not violating the threshold)
 - 3.2. 1 bonus point for finding more than 2 unique failures
 - 3.3. 0 points if the amount of test cases in the covering array is larger than 250
 - 3.4. 0 points if it becomes apparent that the covering array has been tampered with outside of the ACTS tool
 - 3.5. **No matter whether a correct solution was submitted, you get 0 points if the relevant CSV and XML files are not submitted, too**
4. Up to 2 points for Task 3
 - 4.1. Up to 2 points for correctly and completely answering the questions
 - 4.2. 0 points if **it is not explained how you achieved your answer or if you haven't provided the necessary files/marked combinations to verify your result**

Appendix 1

T (strength)	Algorithm	No. of generated test cases	No. of failures (total, including duplicates)	Set of unique failures (e.g. "Error 1, Error 2")	No. of unique failures	Cumulative No. of unique failures up to strength t
1						
2						
3						
4						
5						
6						
7						
8						
9						

Appendix 2

