Brett Worley
CEG-3110-01

Homework 4: Orthogonal array

In order to create a test plan from an orthogonal array, we need to look back at our equivalence classes based off of the requirements. The orthogonal array method of creating test cases is based off of equivalence partitions and the boundary cases that come from that. Using the orthogonal array method to generate test plans requires more thought as to how to setup the program to actually generate an orthogonal array. While originally looking at how to layout the sets of rules to create the orthogonal array, I had put each rule based on the equivalence partitions into a 9 character password set. This setup was not usable, as the number of test cases created this way was too large and a majority of the cases did not provide good testing or make much sense when looking at the requirements.

Table 1: Equivalence Partitions.

| Less than 9 characters | At least 9 characters and not more than 24 characters | More than 24 characters |
|---|---|---|
| Contains no blank spaces | Contains no blank spaces | |
| Contains insufficient lower case | Contains sufficient lower case | |
| Contains insufficient upper case | Contains sufficient upper case | |
| Contains insufficient numbers | Contains sufficient numbers | |
| Contains insufficient special characters | Contains sufficient special characters | |
| Contains a identical five-character substring | Does not contain a identical five-character substring | |

From our previous Equivalence classes, we can make the following border cases. These border conditions are what we will use to setup rules for our orthogonal array generator in order to create usable test cases to produce a good test plan.

1. Less than 9 characters
2. 9 characters
3. 24 characters
4. More than 24 characters
5. Contains insufficient lower case
6. Contains sufficient lower case
7. Contains insufficient upper case
8. Contains sufficient upper case

9. Contains insufficient numbers
10. Contains sufficient numbers
11. Contains insufficient special characters
12. Contains sufficient special characters
13. Contains a identical five-character substring (Similar)
14. Does not contain a identical five-character substring (Not similar)

By using these border conditions, we are able to generate generic orthogonal array test cases. The orthogonal arrays will be based on the setup shown in table 2. Using a program to generate the orthogonal arrays, I created test plans that ran on different numbers of strength for pairwise testing used to create orthogonal arrays. I tested n1, n2, n3, ,6, and n7 and made separate test plans for each just to see how many test cases each plan contained. The n1 test plan consisted of 4 test cases. The n2 plan consisted of 11 test cases. The n3 plan consisted of 27 cases. The n6 plan consisted of 184 cases. The n7 plan consisted of 256 cases. While this method produces test cases that may not be thought of, it can generate some odd test plans. Such as in the n1, n2 test plans, which does not contain a test case that actually passes the program based on the requirements. The n6 n7 generated plans produces at least one test case that passes the program.

Table 2: Order of Orthogonal array results.

| Number of characters | # of characters |
|---|---|
| Contains a space | True/False |
| Sufficient Lower case | True/False |
| Sufficient Upper case | True/False |
| Sufficient numbers | True/False |
| Sufficient Special characters | True/False |
| Similar | True/False |