Wyatt Kormick
CSCI 5802
Reading 1
2/12/18

The problem with testing a software for every configuration, or even for representative values in the Cartesian product of sets of input partitions, is that the actual number of things to test grows so large, that to do it would take an unimaginable amount of time. The article expands upon this problem, presents various versions of a potential solution called "Combinatorial Interaction Testing" (CIT), shows a few problems that these versions might have, and potential solutions to those problems. CIT models the software under test and its inputs. The inputs are put into factor-value pairs, where the factor is the parameter itself, and the value is the assignment to the parameter. A sample of these inputs are taken and a covering array is made for them. A covering array is a list of value configurations for each factor where each combination of factor-values for every combination of some amount of factors appears at least once. This method ends up drastically cutting down the amount of inputs to test without sacrificing too much coverage. This method can be used to quickly find the factor-value pair combinations that can lead to faults in the system.

A problem with this method is something called "masking". What this means is that if one configuration has a fault, then none of the combinations in that configuration can be said to be covered. The usage of the system gets limited by a configuration, and then we are uncertain whether or not parts of the system actually are being tested. The article proposes several solutions to the problem. One of these is partitioning the configuration into subsets of factor-values. By repeatedly doing this, eventually the faulty factor-value pair can be isolated and found. Overall, the system seems to work well for testing a systems inputs. Whatever problems it has, there is a variation of the process that solves it. With the information presented here, this seems like a useful way to come up with a test suite.