Wyatt Kormick

CSCI5802

Reading 2

2/26/18

When compared to a team of professionals writing tests for a system, model checking programs were able to come up with more potential points of fault occurrence. These points often were in places that would be difficult to test for, places that required some very specific sequence of things to happen. Abstracting the software under test in the form of a finite state model has the benefit of being simpler to test all the abstracted states. If the abstraction actually models the system properly, a property that the model has should hold true for the system as well.

The main problem with checking a system's model using such programs is that they cannot be overly complex. There are different model checking programs for different types of complexities, but as of this publication, there isn't one that can deal with all of them. The problem with that is that many complexities in the system under test have to get abstracted away. Abstracting away complexities such as a large state space or value representations give us yet another thing that we are not testing accurately for faults. Advances that could address these limitations would have to be more complex machines to test these. Systems that can accurately represent such complexities without sacrificing too much speed or memory would be a boon to the software engineering community. Modern GPUs for example are a recent advancement that handle floating point values much better than most CPUs.