COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

DISSERTATION DEFENSE

DENIS BUENO

Software Model Checking with Uninterpreted Functions

Wednesday, December 9, 2020
1:00 PM
Virtual (Register in advance)

ABSTRACT: Software model checkers attempt to algorithmically synthesize an inductive proof that a piece of software is safe. Such proofs are composed of complex logical assertions about program variables and control structures, and are computationally expensive to produce.

Our unifying motivation is to increase the efficiency of verifying software control behavior despite its dependency on data. Control properties include important topics such as mutual exclusion, safe privilege elevation, and proper usage of networking and other APIs. These concerns motivate our techniques and evaluations.

We reduce this cost by integrating an efficient abstraction procedure based on the logic of equality with uninterpreted functions (EUF) into the core of a modern model checker. Our checker, called euforia, targets control properties by treating a program's data operations and relations as uninterpreted functions and predicates, respectively. This reduces the cost of building inductive proofs, especially for verifying control relationships in the presence of complex but irrelevant data processing. We show that our method is sound and terminates. We provide a ground-up implementation and evaluate the abstraction on a variety of software verification benchmarks.

We show how to extend this abstraction to memory-manipulating programs. By judicious abstraction of array operations to EUF, we show that we can directly reason about array reads and adaptively learn lemmas about array writes leading to significant performance improvements over existing approaches. We show that our abstraction of array operations completely eliminates much of the array theory reasoning otherwise required. We report on experiments with and without abstraction and compare our checker to the state of the art.

Programs with procedures pose unique difficulties and opportunities. We show how to retrofit a model checker not supporting procedures so that it supports modular analysis of programs with non-recursive procedures. This technique applies to euforia as well as other logic-based algorithms. We show that this technique enables logical assertions about procedure bodies to be reused at different call sites. We report on experiments on software benchmarks compared to the alternative of inlining all procedures.

CHAIR: Prof. Karem Sakallah