Kang Gui
Ph.D.
Computer Engineering
Major Professor: Suraj Kothari

Proving Safety Properties of Software

Abstract:
The use of software is pervasive in areas as diverse as aerospace, automotive, chemical processes, civil infrastructure, energy, health-care, manufacturing, transportation, entertainment, and consumer appliances. Our safety, security, and economy are now closely linked to the reliability of software.

This research is about a technique to prove event-based safety properties of program. A safety property is defined in terms of event traces. An event trace is associated with an execution path and it is the sequence of events that execute on the path. Each event is identified with a program statement or a block of statements. Particularly, this research has been focused on one type of problem that follows one type of safety property we call matching pair (MP) property.

Memory leaks, asymmetric synchronization, and several other defects are examples of violation of the matching pair property. The property involves matching between two types of events on every execution path. We present a practical method to validate the MP property for large software. The method is designed to address the challenges resulting from the cross-cutting semantics and presence of invisible control flow. The method has two phases: the macro phrase and the micro phrase. The macro analysis phase incorporates important notions of signature and matching pair graph (MPG). Signatures enable a decomposition of the problem into small independent instances for validation, each identified by a unique signature. The MPG(X) defines for each signature X, a minimal set of functions to be analyzed for validating the instance. The microanalysis phase produces the event traces graph representing all the relevant execution paths through the functions belonging to a MPG(X). A fast and accurate analysis of large software is possible because the macro analysis can exactly identify the functions that need to be analyzed and the micro analysis further greatly reduce the amount of execution branches in the remaining functions by creating event trace graph (ETG). We applied macro level analysis on 8 versions of Linux kernels spanning for 3 years, which results more than 50% function reduction. We further calculated ETGs for all functions related to MP properties for 3 versions of Linux which results additional 75% reduction w.r.p to number of statements. More than 90% Linux mutex MP cases are proved automatically by our method which is never possible before. For remaining cases, a compact sequence of governing conditions deciding the results are provided for each case for further investigation.