Abstract

Compilers form an integral component of the software development ecosystem. Compiler writers must understand the specification of source and target languages to design sophisticated algorithms that transform programs while preserving semantics. Unfortunately, compiler bugs in mainstream compilers are common. Compiler bugs can manifest as crashes during compilation, or, much worse, result in the silent generation of incorrect programs. Such mis-compilations can introduce subtle errors that are difficult to diagnose and generally puzzling to software developers.

The talk will describe the problems in developing peephole optimizations that perform local rewriting to improve the efficiency of LLVM code. These optimizations are individually difficult to get right, particularly in the presence of undefined behavior; taken together they represent a persistent source of bugs. The talk will present Alive, a domain-specific language for writing optimizations and for automatically either proving them correct or else generating counterexamples. A transformation in Alive is shown to be correct automatically by encoding the transformation into constraints, which are automatically checked for validity using a Satisfiability Modulo Theory (SMT) solver. Furthermore, Alive can be automatically translated into C++ code that is suitable for inclusion in an LLVM optimization pass.

Alive is based on an attempt to balance usability and formal methods; for example, it captures but largely hides the detailed semantics of three different kinds of undefined behavior in LLVM. We have translated more than 300 LLVM optimizations into Alive and, in the process, found that eight of them were wrong. I will conclude the talk highlighting the lessons learned and the challenges in incorporating lightweight formal methods in the tool-chain of the compiler developer.

Bio

Santosh Nagarakatte is an Assistant Professor of Computer Science at Rutgers University. He obtained his PhD from the University of Pennsyl-
vania. His research interests are in Hardware-Software Interfaces spanning Programming Languages, Compilers, and Computer Architecture. His papers have been selected as IEEE MICRO TOP Picks papers of computer architecture conferences in 2010 and 2013. He has received the NSF CAREER Award in 2015, PLDI 2015 Distinguished Paper Award, and the Google Faculty Research Award in 2014 for his research on incorporating lightweight formal methods for LLVM compiler verification.