Practical formal techniques and tools for developing LLVM's peephole optimizations

David Menendez
Dept. of Computer Science

10/3/2017 at 12:30 pm
CoRE A (301)

Abstract

Peephole optimizations are local transformations which perform algebraic simplification to improve performance, reduce code size, or canonicalize code before it is presented to other stages of a compiler. They are a common source of compiler errors. This dissertation presents Alive, a domain-specific language for specifying peephole optimizations in LLVM, and the Alive-NJ toolkit, which automatically checks the correctness of integer- and floating point-based optimizations. The latter are particularly challenging, due to ambiguities in LLVM's semantics. The correctness conditions for Alive optimizations are encoded as constraints, which are checked using a satisfiability modulo theories (SMT) solver. Alive optimizations can also be automatically translated into C++ code suitable for inclusion in LLVM. Incorrect optimizations can frequently be corrected by strengthening their preconditions. The dissertation discusses Alive-Infer, a data-driven method for synthesizing preconditions which make an optimization correct. It addresses the challenges of generating examples in a static setting, learning new predicates through enumeration, and assembling these predicates into a precondition. Finally, the high-level view of LLVM's peephole optimizer provided by Alive allows for analysis of the optimizer as whole. In particular, the Alive-Loops toolkit analyzes sequences of transformations, detects whether they can cause compiler non-termination during optimization, and generates concrete input programs demonstrating this non-termination.

Defense Committee: Prof. Santosh Nagarakatte (Chair), Prof. Ulrich Kremer, Prof. Thu Nguyen, Prof. Rajeev Alur (University of Pennsylvania)