Black-box Equivalence Checking across Compiler Transformations

Author: Manjeet Dahiya
Computer Science and Engineering, IIT Delhi

Abstract

Equivalence checking is an important building block for program synthesis and verification. Design of an equivalence checker is dependent on the application; program synthesis tools like superoptimizer demand that the underlying equivalence checker should perform the required equivalence checks in a black-box manner, i.e., without requiring the knowledge of the individual constituent transformation passes. This thesis presents techniques for black-box equivalence checking across compiler optimizations and across power environments.

In the first part, we present a technique for black-box equivalence checking across compiler optimizations. Unlike previous work on translation validation, our technique works across multiple composed transformations and does not employ a pass-by-pass approach. Our technique supports undefined behaviour related optimizations, and we are the first to handle undefined behaviour related optimizations in equivalence checking for programs with loops. We test our checker with the optimizations produced by multiple modern compilers, and our results are comparable to that of previous work on translation validation, albeit in a black-box setting.

In the second part, we discuss equivalence checking across power environments. Intermittent programs, which are are transiently powered, keep checkpointing the program state to a persistent memory, and on power failures, the programs resume from the last executed checkpoint. An intermittent program is usually automatically generated by instrumenting a given continuous program (continuously powered). The behaviour of the continuous program should be equivalent to that of the intermittent program under all possible power failures. We present a technique to automatically verify the correctness of an intermittent program with respect to its continuous counterpart. We present a model of intermittence to capture all possible scenarios of power failures and an algorithm to automatically find a proof of equivalence between a continuous and an intermittent program.