Compiler Testing via a Theory of Sound Optimisations in the C11/C++11 Memory Model

Robin M., Pankaj P., Francesco Z.

Presented by
Akshay Gopalakrishnan

December 15, 2021

Introduction

Testing sequential programs to hunt for compiler bugs is well established at this time. However, testing concurrency bugs still remains a hard problem. This is majorly due to ill understood specification of the memory consistency model, and lack of information on the various transformations that preserve concurrent program semantics. This work designs a strategy to reduce the complexity of finding concurrency bugs in C11/C++11 via testing sequential code. The source code trace is compared with that of the final end code trace. Previous theoretical work establishing soundness of various local transformations in concurrent execution is utilized to achieve this.

Example of Compiler optimisation

Main idea of testing using traces: Another Example

Overview of C11 memory model

Useful terminology used

Soundness of Optimisations

Using above results for testing

Results

Thank you

Questions?