Correct Compilation of Relaxed Memory Concurrency

PhD Thesis
of
Soham Chakraborty

Presented by
Akshay Gopalakrishnan

July 26, 2021

Introduction

- Relaxed Memory Concurrency (shared memory Concurrency)
- Propose a Relaxed Memory Model based on Event Structures.
- Test the programmibility factors of the model.
- Prove Compilation correctness to various architectures.
- Validate program transformations in LLVM.

Shared Memory Concurrency

Multicore programming. Programs run accross many cores in parallel. Sharing information when needed using a common shared memory. Intuitively, interleaving semantics relied on to reason about multicore programs.

Relaxed Memory Concurrency

Hardwares typically have features which violate interleaving semantics. Gains tremendous performance benefits doing so. Relaxing the interleaving semantics gives rise to more possibilities of program output.

Hardware Memory Models (HMM)

Hardware typically use Read/Write buffers, Cache systems and Speculation which gives rise to relaxed behaviors. This thesis presents three of such hardwares:

- x86-Total Store order.
- PowerPC.
- ARM (v7/v8).

HMM: x86-TSO

HMM: PowerPC

HMM: ARMv7/v8

Software Memory Models (SMM)

Programming languages at the software layer also need a relaxed memory model to be able to use it at the software layer. Relaxation of interleaving semantics open up for doing program transformations for performance. Model needs to support portability of code so that it can run on multiple architectures.

SMM: Data Race Free 0 (DRF0)

SMM: Java

SMM: C11

Additional Problems at the SMM layer

Explaining Behaviors in Relaxed Memory Models

Operational Approach

Transformational Approach

Axiomatic Approach