Towards Transformational Specification of Relaxed Memory Models

Presentation by Akshay Gopalakrishnan
Supervised by Clark Verbrugge

November 23, 2021

Introduction

Intro

Almost every system we use for our day-to-day lives rely on concurrent computation. Right from our mobile phones to our personal desktop which can seemingly perform multiple tasks at the same time, concurrent computations have become part of our daily life. The amount of performance concurrent computation has given us is something that is near impossible to part with. But an important question remains, can we understand and utilize concurrency at its best for the various domain specific compuatations we intend to do?

What are Memory Models?

Concurrent programs take advantage of *out-of-order* execution. Intuitively, this means that more than one unrelated computations can be done "simultaneously" without having any fixed order in which they should happen. This results in concurrent programs having multiple different outcomes. In terms of concurrenct compuations using shared memory, the possible outcomes are described by a *memory consistency model*.

The inclination towards Sequential Reasoning

Despite the elaborate different ways in which we can think of writing programs that leverage the *out-of-order* notion, most programmers are accustomed to reasoning about computations sequentially. Sequential Consistency(SC), which was a memory model first formulated by Lamport et al. [**?**], gives programmers this exact sequential reasoning for their programs running in a multiprocessor environment.

Current Hardwares - the thrist for performance

Though SC seems to be a very intuitive way to reason about programs using shared memory, it does not reflect how different processors do their computation. Hardwares in today's era have multiple features such as caches, read/write buffers, speculative execution, etc. which are designed to give us the performance we quite often take for granted these days. Usage of such features however, does not respect SC reasoning.

Example: Message Passing

Example: Store Buffering

Example: Load Buffering

Example: Non-Multi-Copy Atomic Behavior

Common Problems Associated with Relaxed Memory Models

- Informal specifications.
- Correctness of Compiler Mappings.
- Correctness of Program Transformations.
- Robustness Analysis.
- Compositional Correctness.
- Verification.

Informal Specifications

Correctness of Compiler Mappings

Correctness of Program Transformations

Robustness

Compositional Correctness

Verification

Constructing models using program transformations: Goal

Idea

Simplicity

Possible Advantages - Theoretical

Compiler Correctness

Robustness Conditions

Compositional Correctness

Elements to define models

Constructing base model

Establishing Equivalence

Challenges: Part 1

Challenges: Part 2

Challenges: Part 3

Challenges: Part 4

Challenges: Part 5

Misc

Thank you

Questions?