Explaining Relaxed Memory Models with Program Transformations

Ori Lahav and Viktor Vafeiadis

Presented by
Akshay Gopalakrishnan

September 25, 2021

Introduction

- Uniprocessor semantics respects sequential consistency.
- Mutliprocessor has more behaviors that programs can exhibit.
- They are either justified by hardware features such as Load/Store buffers, speculation, etc.
- Or they can be justified by the program begin optimized in various passes by a compiler.
- Semantics of such behaviors however, are not specified using such intuition, rather, per-execution or event-structure based axiomatic models, or more non-trivial operational models.
- This paper investigates upto what extent such memory consistency models can be specified using program transformations.

Contributions

- TSO can be explained by W-R reordering and Read-after-Write elimination over SC.
- RA cannot be specified in the same manner - counter example to show this.
- A substantial subset of POWER can be specified in a similar way over a stronger model of power - SPOWER
- ARM however, cannot be done the same way.
- Advantage of using this to simplify compilation correctness proofs.

Axiomatic Model Definitions

Sequential Consistency (SC)

Total Store Order (TSO)

TSO Equivalence

TSO is precisely categorized by write-read reordering and read-after-write elimination over SC.

Theorem 1

A plain execution G is TSO-consistent iff $G \longmapsto^*_{TSO} G'$ for some SC-consistent execution G'.

where $\longmapsto^*_{TSO}$ implies we either do write-read reordering or read-after-write elimination.

Proof elements

Proposition 1

If $G \longmapsto^{*}_{TSO} G'$ and G' is TSO consistent, then so is G.

Proposition 2

If G is TSO-consistent execution, then so is $RemoveWR(G, a, b)$.

Proposition 3

If G is TSO-consistent execution, then so is $ReorderWR(G, a, b)$ if $\langle a, b \rangle \notin mo; rf$.

Proof elements cntd

Proposition 4

Suppose G is TSO-consistent but not SC-consistent, then $G \longmapsto^*_{TSO} G'$ for some TSO-consistent execution G'.

Release Acquire (RA)

Why RA cannot be described the same way

Why RA cannot be described the same way

Similar Analysis for POWER and ARMv7/v8

Advantage of Transformational Models: Compiler Correctness

Consider $[[P]]_M$ to be behaviors of program P under memory model M. Assume we have a compilation scheme from source C to target A (code gen phase: direct mapping of instructions). Let M_C and M_A be their respective memory models. Then compiler correctness requires:

$$\forall P_C.[[compile(P_C)]]_{M_A} \subseteq [[P_C]]_{M_C}.$$

Compiler Correctness simplified using results in paper

Theorem 1 and 2 provide us with the following property:

$$\forall P_C.[[compile(P_C)]]_{M_A} \subseteq \{[[P'_A]]_{SM_A} | P'_A.compile(P_C) \longmapsto^*_{M_A} P'_A\}.$$

where SM_A is the stronger memory model than M_A (like SC for TSO in Theorem 1).

Simplified proof requirements

Using above info, compiler correctness boils down to two conditions.

- Compilation should be correct for the strong model SM_A.
- There should be a set of source program transformations sound for source model M_C and it should capture all target transformations from a compiled program.

Thank you

Questions ?