Robustness Between Weak Memory Models

Soham Chakraborty

Presentation by
Akshay Gopalakrishnan

September 13, 2021

Introduction

- Robustness is a property that a program can have given two memory models.
- A program executing under a memory model K is Robust against memory model M if all consistent executions under K are also consistent under M.
- This property is useful in ensuring safe portability of programs across different platforms.
- However, as far as memory models are concerned, Robustness is maintained only against Sequential Consistency (SC).
- This paper analyzes robustness against weaker memory models such as x86-TSO, ARMv7 and ARMv8.
- Robustness conditions are established against each of these models.
- An analysis tool is also developed to ensure a program is Robust or if not, adding fence instructions to ensure Robustness.

Examples of Robustness

As an example, consider robustness of programs running on ARM against x86.

$$\begin{array}{l|l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array} \quad \text{(SB)} \qquad \begin{array}{l|l} a = X; & b = Y; \\ Y = 1; & X = 1; \end{array} \quad \text{(LB)}$$

In the above figure, program SB's outcome $a = 1 \wedge b = 1$ is allowed by both x86 and ARM. But for the program LB, the outcome $a = 1 \wedge b = 1$ is allowed by ARM but not by x86. Thus, program SB is x86 robust while LB is not.

Robustnes Definition

A program P is M-K robust if all its K-consistent executions are also M-consistent.

So for our example above:

- SB is x86-ARM robust.
- LB is not x86-ARM robust.

Axiomatic Specification of Memory Consistency Models

- Define binary relations (mainly partial orders) between program constructs Read/Write/RMW/Fence.
- Per-execution semantics.
- Specify axioms based on relations, which are acyclic/irreflexivity conditions on execution graphs.

Preliminary Definitions of Binary Relations

Binary relations:

- po - Program order per thread.
- rf - Between a read and the write from which its read value comes.
- co - Order between writes/updates to same location.
- rmw - Order between a Read and a Write which form an Atomic update (read-modify-writes).
- fr - $rf^{-1}; co$ read-happens before (also abbreviated as reads-before).
- eco - $rfe \cup coe \cup fre$ called as extended coherence order (take all the external preliminary relations apart from po).
-

SC and X86-TSO

- $po \cup rf \cup fr \cup co$ acyclic.
- $rmw \cap fre; coe = \phi$.

x86-TSO

Some additional binary relations (derived)

- xpo - $((W * W) \cup (R * W) \cup (R * R)) \cap po$ (these events must be ordered)
- imp - $po; [dom(rmw) \cup F] \cup [codom(rmw) \cup F]; po$
- xhb - $xpo \cup imp \cup rfe \cup fr \cup co$

x86 is then defined as

- $po \cup rf \cup fr \cup co$ acyclic.
- $rmw \cap fre; coe = \phi$.
- xhb acyclic.

Robustness Conditions: Intuition

SC-x86TSO Robustness Condition

Checking and Enforcing Robustness

- Take the Control Flow Graph of the program.
- Build a Memory-access pair graph (MPG) capturing two important edge relations (eco and epo).
- If MPG has a cycle, check each access pair whether ordered, as per a condition based on M-K Robustness.
- If every pair is ordered, program is M-K Robust.
- Else, insert appropriate fences between memory access pairs that are not ordered.

Experimental Evaluation

Thank you

Questions?