Robustness Between Weak Memory Models

Soham Chakraborty

Presentation by
Akshay Gopalakrishnan

September 19, 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

SC with the above relations can be defined as:

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

x86-TSO

Some additional binary relations (derived)

- xpo - $((W * W) \cup (R * W) \cup (R * R)) \cap po$ preserved program order.
- imp - $po; [dom(rmw) \cup F] \cup [codom(rmw) \cup F]; po$ implied order.
- xhb - $xpo \cup imp \cup rfe \cup fr \cup co$ x86 happens-before.

x86 is then defined as

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

Robustness Conditions: Intuition

The authors tackle only SC, x86-TSO, ARMv7 and ARMv8 models. All of the above models respect coherence. If for instance robustness fails to hold, then we have some K-consistent execution to violate an axiom of memory model M. Since our axioms are all acyclic/irreflexivity conditions, this implies the execution graph has some form of cycle. Note that, if the cycle contains no po edges, then with the remaining edges (rfe, fre, coe), the cycle also results in coherence violation. Thus, the cycle must contain po edges along with the union of the above (now termed as eco). This set of po edges are defined as extended program order $epo = po \cap (codom(eco) \times domeco)$. The robustness condition now between each pair of memory model revolves around this newly defined order.

SC-x86TSO Robustness Condition

The intuition behind the following theorem is that, we know for certain because of coherence that a cycle if introduced must have po an exclusive part. Thus, it is sufficient to place restrictions on this set of po to belong to a fixed set to ensure no cycle can exist.

Theorem

A program P is M-K Robust if in all it's K-consistent executions X, epo $\subseteq$ R where R is defined for each pair of memory models in question.

For SC-x86TSO robustness R is defined as

$$R = xpo \cup po_l \cup imp; po^?. \tag{1}$$

SC-x86TSO Robustness Condition

The intuition behind the following theorem is that, we know for certain because of coherence that a cycle if introduced must have po an exclusive part. Thus, it is sufficient to place restrictions on this set of po to belong to a fixed set to ensure no cycle can exist.

Theorem
A program P is M-K Robust if in all it's K-consistent executions X, epo ⊆ R where R is defined for each pair of memory models in question.

For SC-x86TSO robustness R is defined as

$$R = xpo \cup po_{\mid} \cup imp; po^{?}. \quad (1)$$

The intuition behind SC-x86 robustness condition is as follows:

- xpo orders all R-R, R-W and W-W events. These orders does not actually represent the main weakening that makes it TSO.
- $po_{\mid}$ If the whole program is surrounding just one memory action, then the program under TSO/SC is the same.
- $imp; po^{?}$ Any fence between W-R would imply that R will read from the main memory, just like SC semantics.

The above three are not the orders that make a Read have a choice to read from its own Write Buffer. This would be different from SC and thus will not be SC-x86 Robust. Thus, we require epo to be in the above union of 3 relations.

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

Intrpreting the Results. The (SC-K) entries in the tables are of the form (a|b(✓/✗) c ⟨ d) where

- 'a': number of fences required by naive scheme.
- 'b': number of existing fences in the program.
- 'c': number of fences inserted by proposed scheme.
- '✓/✗' symbol denotes if a program is M-K robust or not.
- 'd': time taken by the robustness pass in seconds.

Experimental Evaluation Table

Prog.	SC-x86 result	SC-x86 ⟨sec	Trencher result	Trencher ⟨sec
Barrier	6|0✗2	⟨0.005	✗2	⟨0.004
Dekker-TSO	20|4✓0	⟨0.002	✓0	⟨0.007
Peterson-SC	14|0✗2	⟨0.004	✗2	⟨0.013
Lamport-SC	17|0✗4	⟨0.019	✗4	⟨0.107
Spinlock	14|0✓0	⟨0.004	✓0	⟨0.007
Ticketlock	12|0✓0	⟨0.004	✓0	⟨0.006
Seqlock	7|0✓0	⟨0.004	✓0	⟨0.582
RCU-offline	33|4✗3	⟨0.038	✗-	⟨0.246
Cilk-TSO	22|2✓0	⟨0.011	✗0	⟨2.039
Cilk-SC	22|0✓0	⟨0.010	✓2	⟨6.322

Author makes note of Cilk-TSO specifically. Trencher reports violation due to not including po_l in condition check.

Thank you

Questions?