Weak Memory Concurrency-II

Soham Chakraborty

02.03.2022

Outline

Axioms & Properties

- Coherence
- Atomicity

Relaxed Memory Models in Programming Languages

Relaxed Memory Models in Architectures

Coherence

SC per location

(poloc $\cup$ rf $\cup$ fr $\cup$ mo) is acyclic

Examples: coherence violations

$$X = 1;$$
$$X = 2;$$
$$a = X; \ // \ 1$$

$$a = X; \ // \ 1$$
$$X = 1;$$

$$X = 1; \quad \| \quad a = X; \ // \ 2$$
$$X = 2; \quad \| \quad b = X; \ // \ 1$$

$$X = 1; \qquad \| \quad X = 2;$$
$$a = X; \ // \ 2 \quad \| \quad b = X; \ // \ 1$$

Atomicity

Atomicity: $\mathsf{rmw} \cap (\mathsf{fr}; \mathsf{mo}) = \emptyset$

Examples: Atomicity violations

$$\begin{array}{c} X = 0; \\ CAS(X, 0, 1); \parallel CAS(X, 0, 1); \end{array}$$

Both CAS operations cannot be successful

Release-Acquire Consistency

All writes are release and all reads are acquire accesses

Follows (sc-per-loc) and (atomicity)

Reordering restrictions: WW, RR, RW (same as TSO)

Example: Allowed behaviors (same as TSO)

$$X = Y = 0;$$

$$\begin{array}{l||l} X = 1; & a = Y; \ // \ 1 \\ Y = 1; & b = X; \ // \ 0 \end{array}$$

$$X = Y = 0;$$

$$\begin{array}{l||l} X = 1; & Y = 1; \\ a = Y; \ // \ 0 & b = X; \ // \ 0 \end{array}$$

Release-Acquire Consistency

Reordering restrictions: WW, RR, RW (same as TSO)

Allows non-multicopy atomicity unlike TSO

Example:

$$X = Y = 0;$$

$$X = 1; \;\Big\|\; \begin{array}{l} a = X; \\ b = Y; \end{array} \;\Big\|\; \begin{array}{l} c = Y; \\ d = X; \end{array} \;\Big\|\; Y = 1;$$

Outcome $a = c = 1, b = d = 0$ is allowed in RA but not in TSO

Release-Acquire Consistency

Axioms

$(\mathsf{poloc} \cup \mathsf{rf} \cup \mathsf{fr} \cup \mathsf{mo})$ is acyclic (sc-per-loc)

$\mathsf{rmw} \cap (\mathsf{fr}; \mathsf{mo}) = \emptyset$ (atomicity)

$\mathsf{hb}; \mathsf{eco}^?$ is irreflexive where (RA)

- $\mathsf{hb} \triangleq (\mathsf{po} \cup \mathsf{rf})^+$
- $\mathsf{eco} = (\mathsf{rf} \cup \mathsf{fr} \cup \mathsf{mo})^+$

Mappings: RA to TSO

$W \rightsquigarrow W$, $R \rightsquigarrow R$, $\mathit{CAS} \rightsquigarrow \mathit{CAS}$, $F \rightsquigarrow F$

Mapping Correctness: Suppose a program P in RA is mapped to P' in TSO following the above mapping scheme. For each TSO-consistent execution of P' there exists an RA-consistent execution of P having same behavior.

- Behavior: Final values in the shared memory locations

C/C++ Concurrency

Non-atomic accesses

Relaxed accesses

Acquire, release accesses and fences

SC accesses and SC fence

Formal model is known as C11

Relations

Synchronization relation is established by

- Release acquire accesses
- Relaxed accesses and fences

Happens-before: $\mathsf{hb} = (\mathsf{po} \cup \mathsf{sw})^{+}$

Data Race

a and b is a data race (on non-atomics) when

- a and b are concurrent (not related by hb)
- Access same memory location
- Atleast one of a or b is non-atomics

A consistent execution with data race on non-atomic
$\implies$ the behavior of the program is undefined

$$X = 0$$

$$X_{\text{na}} = 1; \parallel a = X_{\text{acq}};$$

$a > 1$ is possible in C/C++

OOTA: out-of-thin-air behavior

$$X = Y = 0$$

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

$$X = Y = 0$$

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

Undesirable: C11 allows $a = b = 1$ in both programs

Desirable: Allow $a = b = 1$ in (LB), but forbid in (LBDep)

Architecture Memory Models

Examples: x86, ARMv8, ARMv7, Power

Dependency orders accesses (unlike C11)

- (LB) and (LBDep) programs

Fences are different

Data race has no undefined behavior

Discussions

Why different memory models?

Role of compilers

Considerations for a new memory model

Analysis tools

References

Mathematizing C++ Concurrency.
Mark Batty, Scott Owens, Susmit Sarkar, Peter Sewell, and Tjark Weber.
In POPL 2011

Chapter 2 (Background)
Correct Compilation of Relaxed Memory Concurrency.
Soham Chakraborty
`http://plv.mpi-sws.org/soham/thesis/Thesis-Chakraborty.pdf`