Weak Memory Concurrency-II

Soham Chakraborty

27.02.2023

Outline

Semantics

Reasoning about Properties

- Coherence
- Atomicity

Concurrency and compilation

Semantics Models for Concurrency

Transformation+Interleaving

Operational semantics

Axiomatic semantics

Axiomatic Semantics

Program is a set of *consistent* execution

Execution = events + relations

- events represent accesses on shared memory locations or fences
- Events are related by binary Relations

Axioms enforce constraints on an execution

Memory model = set of axioms

Given a memory consistency model M and an execution X, X is M-consistent if X satisfies all the axioms of M

Event

event =
$\langle$unique identifier, thread id, operation, order, location, value$\rangle$

Examples

A release write access writes value v on location X

- $\mathrm{St}_{\mathrm{rel}}(X, v)$

A acquire read access reads value v from location X

- $\mathrm{Ld}_{\mathrm{acq}}(X, v)$

Relations

Each relation relate two events

Example
Relation Program-Order (po): captures syntactic order of the events

Relation reads-from (rf):
If a read r reads-from a write w then $(w, r) \in \text{rf}$

Constraints:

- w and r access same memory locations
- Written value of w = read value of r

Modification order (mo): orders write events that access the same memory location

Execution Graph

Execution is represented as a graph

Nodes are events and edges are relations

Axioms are (usually) acyclicity conditions

Example: No read access reads from a later write

More Relations

Relations can be primitive or derived from other relations

From-read(fr): orders a read to a later write on the same location

Derived from rf and mo

$\mathsf{fr} \triangleq \mathsf{rf}^{-1}; \mathit{mo}$

Sequential Consistency

All shared memory accesses are ordered

$(\mathsf{po} \cup \mathsf{rf} \cup \mathsf{mo} \cup \mathsf{fr})$ are acyclic.

Example: Forbid weak execution(s)

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

Outcome $a = b = 0$ are disallowed

Coherence Property

SC per location

A new relation:
poloc: po between events that access the same memory location

$\text{poloc} = \{(a, b) \mid \text{po}(a, b) \wedge a.location = b.location\}$

Coherence: $(\text{poloc} \cup \text{rf} \cup \text{fr} \cup \text{mo})$ is acyclic

Examples: coherence violations

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

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

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

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

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 in these programs)

$$X = Y = 0;$$

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

$$X = Y = 0;$$

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

Release-Acquire Consistency

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

Allows non-multicopy atomicity unlike TSO

Example:

$$X = Y = 0;$$

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

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

Release-Acquire Consistency

New relation

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

Identify the hb relations in earlier examples

Axiomns

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

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

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

$$X = 0$$

$$X_{\mathsf{na}} = 1; \;\|\; a = X_{\mathsf{acq}};$$

$a > 1$ 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)

Correctness of Compilation

Correct transformations in sequential programs may NOT be correct in concurrent programs

Example: Under RA model

$$\begin{array}{c} X = Y = 0; \\ X = 1; \;\|\; a = Y; \\ Y = 1; \;\|\; b = X; \end{array} \qquad \rightsquigarrow \qquad \begin{array}{c} X = Y = 0; \\ X = 1; \;\|\; a = Y; \\ Y = 1; \;\|\; b = X; \end{array}$$

$a = 1, b = 0$ is forbidden $\qquad$ $a = 1, b = 0$ is allowed

Incorrect transformation

Correctness of Compilation

For each consistent execution of the target program there exists a corresponding consistent execution of the source program with same outcome.

Proof Strategy

1. For each consistent execution X of the target program, define an execution X' of the source program.
2. Show that X' is consistent w.r.t the source consistency model
3. Show the behavior of X and X' are same.

Correctness of Compilation

Transformtions:

- Program optimization
- Mapping to architectures

Correct Program Transformations

Reordering transformations: $a; b; \rightsquigarrow b; a;$

Elimination transformations:

- $t = X; t' = X; \rightsquigarrow t = X; t' = t;$ (RAR)
- $X = v; t = X; \rightsquigarrow X = v; t = v;$ (RAW)
- $X = v; X = v'; \rightsquigarrow X = v';$ (OW)

Mapping Schemes

Programming languages primitives to architectures

Additional leading and/or trailing fences

Example

`https://www.cl.cam.ac.uk/~pes20/cpp/cpp0xmappings.html`

Main Challenge: Proving mapping correctness

Example Mappings: RA to TSO

$\mathsf{W}_{\mathsf{rel}} \rightsquigarrow \mathsf{W}$, $\mathsf{R}_{\mathsf{acq}} \rightsquigarrow \mathsf{R}$, $\mathit{CAS} \rightsquigarrow \mathit{CAS}$, $\mathsf{F} \rightsquigarrow \mathsf{F}$

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

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`

`https://www.cse.iitd.ac.in/~soham/COL869/page.html`
Lectures: 30.03.2020 - 24.04.2020