Basic operational semantics of concurrency

Homework assignment

September 21, 2023

Abstract

The purpose of this assignment is to get you familiar with the operational semantics of SC, TSO, and PSO memory models. You have to provide formal definitions of these models and prove several statements about their relationship.

Task 1 (2 points)

Provide definition of operational semantics of the TSO and PSO memory subsystems (you earn 1 point for each). Please find the informal description of the models below. You can use the definition of SC model semantics as a reference (see Fig. 1).

Recall that *memory subsystem* is a *labeled transition system*, $\langle \mathsf{Mem}, \mathsf{Tid} \times \mathsf{Lab}, \rightarrow \rangle$, where:

- Mem is a set of memory states;
- Tid is a set of thread identifiers;
- Lab is a set of labels (see below);
- $\rightarrow \subseteq \mathsf{Mem} \times \mathsf{Lab} \times \mathsf{Mem}$ is a labeled transition relation:

$$m \xrightarrow{\ell} m' \triangleq \langle m, \ell, m' \rangle \in \rightarrow .$$

A label $\ell \in \mathsf{Lab}$ can be one of the following:

- $\texttt{R}(x, v)$ is a read label of value $v \in \mathsf{Val}$ from memory location $x \in \mathsf{Loc}$;
- $\texttt{W}(x, v)$ is a write label of value $v \in \mathsf{Val}$ into memory location $x \in \mathsf{Loc}$;
- $\texttt{U}(x, v_r, v_w)$ is a atomic update label of memory location $x \in \mathsf{Loc}$, reading value $v_r \in \mathsf{Val}$ and writing value $v_w \in \mathsf{Val}$;
- $\texttt{F}$ is a fence label.

Sequential Consistency (SC) is a memory consistency model where any result of executing a multithreaded program can be explained as a sequential execution of some interleaving of threads' instructions. This model forbids reordering of any instructions within same thread. The formal semantics of SC memory subsystem is given on Fig. 1.

$$\mathsf{Mem} \triangleq \mathsf{Loc} \to \mathsf{Val}$$

(a) State definition

$$\frac{m(x) = v}{m \xrightarrow{t:\texttt{R}(x,v)} m}\ \text{Read} \qquad \frac{}{m \xrightarrow{t:\texttt{W}(x,v)} m[x \mapsto v]}\ \text{Write} \qquad \frac{}{m \xrightarrow{t:\texttt{F}} m}\ \text{Fence}$$

$$\frac{m(x) = v_r}{m \xrightarrow{t:\texttt{U}(x,v_r,v_w)} m[x \mapsto v_w]}\ \text{Update}$$

(b) Transition relation

Figure 1: SC memory subsystem

Total Store Ordering (TSO) is a memory consistency model where loads may be executed before earlier stores to different addresses are completed, but stores are executed in program order. This is achieved by supplementing each execution thread with a **store buffer**. Buffered stores are propogated into the main memory in FIFO order. The formal semantics of TSO memory subsystem is given on Fig. 2.

TODO:

(a) State definition

TODO:

(b) Transition relation

Figure 2: TSO memory subsystem

Partial Store Ordering (PSO) is a memory consistency model where loads can be executed earlier stores to different addresses (similarly as in TSO), but also stores to different addresses can be reordered with respect to one another. In other words, stores to different addresses, while residing in the store buffer, can be reordered before they are propogated into the main memory. The formal semantics of PSO memory subsystem is given on Fig. 3.

TODO:

(a) State definition

TODO:

(b) Transition relation

Figure 3: PSO memory subsystem

Task 2 (3 points)

Prove that TSO memory model is weaker than SC.

Formally, you need to show that each valid trace of SC memory subsystem is also a valid trace of TSO memory subsystem.

Definition 1. A trace of labeled transition system $\langle S, L, \to \rangle$ is an alternating sequence of states and labels: $s_1, \ell_1, s_2, \ell_2, \ldots, s_n, \ell_n, s_{n+1}$. It is a valid trace, if each each pair of neighboring states forms a valid transition, that is:

$$s_1 \xrightarrow{\ell_1} s_2 \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_{n-1}} s_n \xrightarrow{\ell_n} s_{n+1}$$

To do so, please use the *simulation method*, discussed on the seminar.

Definition 2. Given two labeled transition systems $\Sigma = \langle S, L, \to \rangle$ and $\mathrm{T} = \langle U, L, \to \rangle$, a relation $R \subseteq S \times U$ is called a *simulation relation* if the following condition met:

$$\forall\ s\ u\ u'\ \ell.\, \langle s, u\rangle \in R \wedge u \xrightarrow{\ell} u' \implies \exists\ s'.\, \langle s', u'\rangle \in R \wedge s \xrightarrow{\ell} s'$$

If there exists such a simulation relation, we also say that Σ *simulates* T.

Theorem 3. TSO memory model is weaker than SC: every valid trace of SC memory subsystem is also a valid trace of TSO memory subsystem.

Proof. **TODO:** □

Task 3 (5 points)

Prove that if a memory fence is inserted after each store, then program will have the same outcome under TSO and SC memory models.

Formally, you need to prove that giving a trace of TSO memory subsystem, where in each thread every store is followed by a fence, there exists a program order preserving reordering of this trace that gives a similar outcome under SC memory subsystem.

Theorem 4. Consider pair of states $m \in \mathsf{Mem}_{\mathsf{TSO}}$ and $\overline{m} \in \mathsf{Mem}_{\mathsf{SC}}$, such that the main memory contents of both states are the equal, and store buffer of TSO state is empty. Suppose that $[t_1 : \ell_1, \ldots, t_n : \ell_n]$ is a sequence of thread ids and labels, satisfying the following condition:

$$\forall\ i.\, \ell_i = \mathtt{W}(x, v) \implies \exists\ j.\, t_j = t_i \wedge \ell_j = \mathtt{F} \wedge \forall k.\, i < k < j \implies t_k \neq t_i.$$

Show that for every m', such that $m \xrightarrow{t_1:\ell_1} \ldots \xrightarrow{t_n:\ell_n} m'$, there exists a reordering of labels $f : \mathbb{N} \to \mathbb{N}$ (*i.e.,* bijective function), which satisfies the following conditions:

- reordering preserves program order:
$$\forall i\ j.\, t_i = t_j \wedge i < j \implies f(i) < f(j)$$
- for a state $\overline{m'}$, such that $\overline{m} \xrightarrow{t_{f(1)}:\ell_{f(1)}} \ldots \xrightarrow{t_{f(n)}:\ell_{f(n)}} \overline{m'}$, the main memory contents of m' and $\overline{m'}$ are equal, and the store buffer of TSO state m' is empty.

Proof. **TODO:** □

References

[1] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.

[2] P. Sewell, S. Sarkar, S. Owens, F. Z. Nardelli, and M. O. Myreen. x86-TSO: A rigorous and usable programmer's model for x86 multiprocessors. *Commun. ACM*, 53(7):89–97, 2010.