HW2 – Formal Memory Models

CS 554, Fall 2025
Due Sep. 5

1 Rules

1.1 SC, operational

$$\text{store}\ \frac{\sigma_i = (\texttt{store}\ \ell\ v; \sigma_i')}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \to (\sigma_1 \parallel ... \ \sigma_i' \ ... \parallel \sigma_n, m[\ell \mapsto v])}$$

$$\text{load}\ \frac{\sigma_i = (r\ \texttt{=}\ \texttt{load}\ \ell; \sigma_i')}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \xrightarrow{R\ r\ m(\ell)} (\sigma_1 \parallel ... \ \sigma_i' \ ... \parallel \sigma_n, m)}$$

1.2 TSO, operational

$$\text{store}\ \frac{\sigma_i = (\texttt{store}\ \ell\ v; s_i', b)}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \to (\sigma_1 \parallel ... \ (s_i', W\ \ell\ v; b) \ ... \parallel \sigma_n, m)}$$

$$\text{propagate}\ \frac{\sigma_i = (s_i, b; W\ \ell\ v)}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \to (\sigma_1 \parallel ... \ (s_i, b) \ ... \parallel \sigma_n, m[\ell \mapsto v])}$$

$$\text{load-mem}\ \frac{\sigma_i = (r\ \texttt{=}\ \texttt{load}\ \ell; s_i', b) \qquad W\ \ell \notin b}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \xrightarrow{R\ r\ m(\ell)} (\sigma_1 \parallel ... \ (s_i', b) \ ... \parallel \sigma_n, m)}$$

$$\text{load-buf}\ \frac{\sigma_i = (r\ \texttt{=}\ \texttt{load}\ \ell; s_i', b) \qquad \text{first } \ell \text{ in } b \text{ is } W\ \ell\ v}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \xrightarrow{R\ r\ v} (\sigma_1 \parallel ... \ (s_i', b) \ ... \parallel \sigma_n, m)}$$

$$\text{fence}\ \frac{\sigma_i = (\texttt{fence}; s_i', \cdot)}{(\sigma_1 \parallel ... \ \sigma_i \ ... \parallel \sigma_n, m) \to (\sigma_1 \parallel ... \ (s_i', \cdot) \ ... \parallel \sigma_n, m)}$$

Note that $\cdot$ means an empty list.

1.3 SC, axiomatic

1. Every operation within a thread is ordered by program order (po).
2. There is a total order (to) on all operations, consistent with po.
3. Each read $R\ \ell\ v$ has the same value as the immediately preceding write to ℓ in to.

1.4 TSO, axiomatic

1. Every operation within a thread is ordered by program order (po), *except* write operations to read operations.
2. There is a total order (tso) on all write operations, consistent with po.
3. There is a partial read-from order (rf) that orders a write $W\ \ell\ v$ before any reads *in other threads* that read its value. Each read reads either the value of the immediately preceding write in the same thread, or the write it has an rf edge to. Furthermore, for each pair of a write $W\ \ell\ v$ and a read $R\ \ell\ v$ that reads from it, there must not be another write to the same location ($W\ \ell\ v'$) ordered between them, or ordered after $W\ \ell\ v$ and earlier in the same thread as $R\ \ell\ v$. Equivalently, we can draw a from-read edge (fr) from each read to the tso-next write to the same location, and there are no cycles in the order $\mathsf{po}'_\ell \cup \mathsf{tso} \cup \mathsf{rf} \cup \mathsf{fr}$ (where po'_ℓ orders all operations on ℓ within the same thread, *including* writes to reads).
4. The combination of all of these orders ($\mathsf{po} \cup \mathsf{tso} \cup \mathsf{rf}$) is acyclic.

2 Problems

This is a written homework with 3 problems, each of which has multiple parts.

1. Consider the following program:

```
store(x, 1)  ||  store(y, 1)
r1 = load(y) ||  r2 = load(x)
```

   (a) Show a sequence of steps this program can take in the operational SC memory model, producing the outcome where `r1` is 0 and `r2` is 1.

   (b) Show a sequence of steps this program can take in the operational TSO memory model, producing the outcome where `r1` is 0 and `r2` is 1.

(c) Draw an execution graph for the axiomatic SC model that justifies the outcome where `r1` is 0 and `r2` is 1.

(d) Draw an execution graph for the axiomatic TSO model that justifies the outcome where `r1` is 0 and `r2` is 1.

2. Suppose we changed the operational TSO model's load-mem rule to remove the requirement $W\ \ell \notin b$.

   (a) Intuitively, how would this change the behavior of the machine?

   (b) Write a concrete program where this change would be visible—i.e., one that can have an outcome under the changed model that isn't possible under the real TSO model.

3. Suppose we changed the axiomatic TSO model to remove the "no intervening write" condition and the fr edges, i.e., we changed rule 3 to simply "There is a partial read-from order (rf) that orders a write $W\ \ell\ v$ before any reads in other threads that read its value. Each read reads either the value of the immediately preceding write in the same thread, or the write it has an rf edge to."

   (a) Intuitively, how would this change the behavior of the model?

   (b) Write a concrete program where this change would be visible—i.e., one that can have an outcome under the changed model that isn't possible under the real TSO model.

   (c) Can you think of a way to make the same change to the operational TSO model? Or, can you explain why it would be impossible to make such a change? (This will be very generously graded—just write out your thoughts.)