

HW2 – Operational Memory Models

CS 554, Fall 2026

Due Sep. 25

1 Rules

1.1 SC operational model

$$\begin{array}{c} \text{store} \frac{\sigma_i = (\text{store } \ell \ v; \sigma'_i)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \rightarrow (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m[\ell \mapsto v])} \\ \\ \text{load} \frac{\sigma_i = (r = \text{load } \ell; \sigma'_i)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \xrightarrow{R \ r \ m(\ell)} (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)} \end{array}$$

1.2 TSO operational model

$$\begin{array}{c} \text{store} \frac{\sigma_i = (\text{store } \ell \ v; s, b) \quad \sigma'_i = (s, b; \ell \mapsto v)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \rightarrow (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)} \\ \\ \text{propagate} \frac{\sigma_i = (s, \ell \mapsto v; b) \quad \sigma'_i = (s, b)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \rightarrow (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m[\ell \mapsto v])} \\ \\ \text{load-mem} \frac{\sigma_i = (r = \text{load } \ell; s, b) \quad \ell \notin b \quad \sigma'_i = (s, b)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \xrightarrow{R \ r \ m(\ell)} (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)} \\ \\ \text{load-buf} \frac{\sigma_i = (r = \text{load } \ell; s, b) \quad \text{last } \ell \text{ in } b \text{ is } \ell \mapsto v \quad \sigma'_i = (s, b)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \xrightarrow{R \ r \ v} (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)} \\ \\ \text{fence} \frac{\sigma_i = (\text{fence}; s, \cdot) \quad \sigma'_i = (s, \cdot)}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \rightarrow (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)} \end{array}$$

Note that $\cdot$ means an empty list.

1.3 SRA operational model

$$\begin{array}{c}
\text{store} \frac{\sigma_i = (\text{store } \ell \ v; s, V) \quad V(\ell) < t \quad (\ell, t) \notin m \quad V' = V[\ell \mapsto t] \quad \sigma'_i = (s, V')}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \rightarrow (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m \uplus (\ell, v, t, V'))} \\
\\
\text{load} \frac{\sigma_i = (r = \text{load } \ell; s, V) \quad V(\ell) \leq t \quad (\ell, v, t, V') \in m \quad \sigma'_i = (s, V \sqcup V')}{(\sigma_1 \parallel \dots \sigma_i \dots \parallel \sigma_n, m) \xrightarrow{R \ r \ v} (\sigma_1 \parallel \dots \sigma'_i \dots \parallel \sigma_n, m)}
\end{array}$$

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) 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 0.

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

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

(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. Consider the following program:

```
store(x, 1) || r1 = load(x)
store(x, 2) || r2 = load(x)
```

- [illegible]