Operational Memory Models: SC, TSO, SRA

December 2, 2025

1 SC

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

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

2 TSO

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

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

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

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

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

Note that $\cdot$ means an empty list.
Key points:

1. The store rule adds writes to the *front* of the buffer.
2. The propagate rule removes writes from the *back* of the buffer.
3. The load-buf rule checks from the *front* of the buffer.

3 SRA

$$\text{store}\ \frac{\text{there is no write to } \ell \text{ with timestamp } t \text{ in } m \qquad V_i(\ell) < t \qquad V_i' = V_i[\ell \mapsto t]}{(m, V_1 ... V_i ... V_n) \xrightarrow{W\ i\ \ell\ v} (m \uplus (\ell, v, t, V_i'), V_1 ... V_i' ... V_n)}$$

$$\text{load}\ \frac{(\ell, v, t, V) \in m \qquad V_i(\ell) \leq t}{(m, V_1 ... V_i ... V_n) \xrightarrow{R\ i\ \ell\ v} (m, V_1 ... (V_i \sqcup V) ... V_n)}$$

Key points:

1. The store rule adds a write to the memory ("message pool") m.
2. In the load rule, a thread can only read a write to ℓ that is at least as recent as the last write to ℓ it synchronized with ($V_i(\ell)$).
3. Each write in memory carries the view V of the thread that wrote it, and when a thread reads a write it adds that view to its own ($V_i \sqcup V$), i.e., it synchronizes with the writing thread.