Operational Memory Models: SC and TSO

October 1, 2025

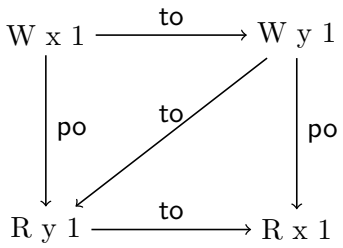
1 SC

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.

Example:

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

Result: r1 = 1, r2 = 1



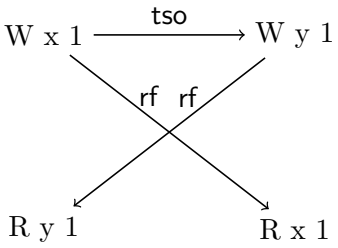
2 TSO

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 either receives such an rf edge, or reads the value of the preceding write to the same location in the same thread. 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.

Examples:

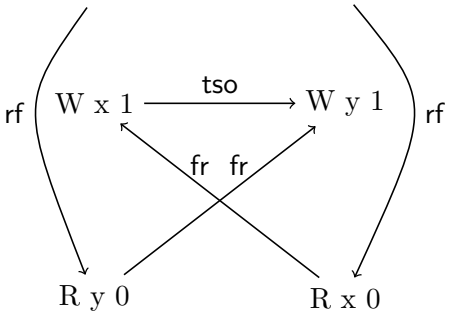
```
store(x, 1)   ||  store(y, 1)
r1 = load(y)  ||  r2 = load(x)
```

Result: `r1` = 1, `r2` = 1




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

Result: `r1` = 0, `r2` = 0



3 RA

1. Every operation within a thread is ordered by program order (po).
2. There is a read-from (rf) edge to each read from the write it reads from (i.e., a write to the same location with the same value).
3. $\mathsf{hb} := \mathsf{po} \cup \mathsf{rf}$ is acyclic.
4. For each location ℓ, there is a modification order mo_ℓ that is a total order on writes to ℓ. If $w_1 \xrightarrow{\mathsf{rf}} r$ and $w_1 <_{\mathsf{mo}_\ell} w_2$, then $w_2 \not<_{\mathsf{hb}'_\ell} r$, where $\mathsf{hb}'_\ell := \mathsf{hb} \cup \mathsf{mo}_\ell$. Equivalently, there is an from-read order fr_ℓ such that if $w_1 \xrightarrow{\mathsf{rf}} r$ and $w_1 \xrightarrow{\mathsf{mo}_\ell} w_2$ then $r \xrightarrow{\mathsf{fr}_\ell} w_2$, and $\mathsf{hb} \cup \mathsf{mo}_\ell \cup \mathsf{fr}_\ell$ is acyclic for each ℓ.

Example:

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

Result: `r1` = 0, `r2` = 0