

Axiomatic Memory Models: SC, TSO, RA

October 19, 2026

First, draw an *event structure*, a graph containing all the reads and writes in the program with operations from the same thread connected by program order (**po**) edges. Then, try to apply the rules of the memory model to complete the event structure into an *execution*.

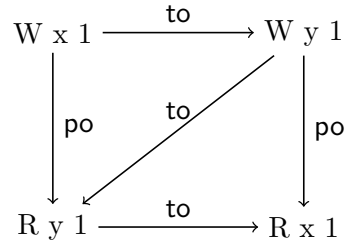
1 SC

1. There is a total order **to** on all operations, consistent with **po**.
2. Each read $R \ell v$ has the same value as the immediately preceding write to ℓ in **to**.

Example:

$\text{store}(x, 1) \parallel \text{store}(y, 1)$
 $\text{r1} = \text{load}(y) \parallel \text{r2} = \text{load}(x)$

Result: $\text{r1} = 1, \text{r2} = 1$



2 TSO

1. The *preserved program order* (**ppo**) includes all **po** edges *except* those from write operations to read operations.
2. There is a total order **tso** on all write operations, consistent with **ppo**.
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 reads-before edge (**rb**) from each read to the **tso**-next write to the

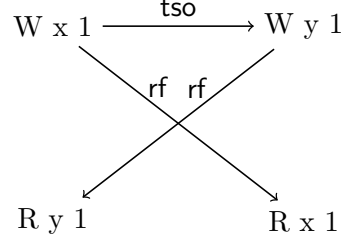
same location, and there are no cycles in the order $\text{po}_\ell \cup \text{tso} \cup \text{rf} \cup \text{rb}$ (where po_ℓ orders all operations on ℓ within the same thread, *including* writes to reads).

4. $\text{hb} := \text{ppo} \cup \text{tso} \cup \text{rf} \cup \text{rb}$ is acyclic.

Examples:

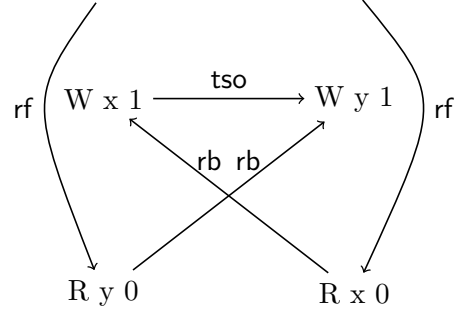
$\text{store}(x, 1) \parallel \text{store}(y, 1)$
 $\text{r1} = \text{load}(y) \parallel \text{r2} = \text{load}(x)$

Result: $\text{r1} = 1, \text{r2} = 1$



$\text{store}(x, 1) \parallel \text{store}(y, 1)$
 $\text{r1} = \text{load}(y) \parallel \text{r2} = \text{load}(x)$

Result: $\text{r1} = 0, \text{r2} = 0$



3 RA

1. 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).
2. $\text{hb} := \text{po} \cup \text{rf}$ is acyclic.
3. “No intervening writes”: For each location ℓ , there is a modification order mo_ℓ that is a total order on writes to ℓ . If $w_1 \xrightarrow{\text{rf}} r$ and $w_1 <_{\text{mo}_\ell} w_2$, then $w_2 \not\prec_{\text{hb} \cup \text{mo}_\ell} r$. Equivalently, there is a reads-before order rb_ℓ such that if $w_1 \xrightarrow{\text{rf}} r$ and $w_1 \xrightarrow{\text{mo}_\ell} w_2$ then $r \xrightarrow{\text{rb}_\ell} w_2$, and $\text{hb} \cup \text{mo}_\ell \cup \text{rb}_\ell$ is acyclic for each ℓ .

Example:

```

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

```

Result: r1 = 0, r2 = 0

