

HW3 – Axiomatic Memory Models

CS 554, Fall 2026

Due Oct. 9

1 Rules

1.1 SC axiomatic model

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**.

1.2 TSO axiomatic model

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 $\mathbf{po}_\ell \cup \mathbf{tso} \cup \mathbf{rf} \cup \mathbf{rb}$ (where $\mathbf{po}_\ell$ orders all operations on ℓ within the same thread, *including* writes to reads).
4. $\mathbf{hb} := \mathbf{ppo} \cup \mathbf{tso} \cup \mathbf{rf} \cup \mathbf{rb}$ is acyclic.

1.3 RA axiomatic model

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

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) Draw an execution graph for the axiomatic SC model that justifies the outcome where **r1** is 0 and **r2** is 1. Clearly indicate the total order on the operations.

- (b) Draw an execution graph for the axiomatic TSO model that justifies the outcome where $\mathbf{r1}$ is 0 and $\mathbf{r2}$ is 1. Clearly indicate the edges that are included in $\mathbf{hb}$.

- (c) Draw an execution graph for the axiomatic RA model that justifies the outcome where $r1$ is 0 and $r2$ is 1. Clearly indicate the edges that are included in hb .

2. Consider the following program:

$$\begin{array}{l} \text{store } x \ 1 \parallel \text{store } y \ 2 \parallel \begin{array}{l} r1 = \text{load } x \\ r2 = \text{load } y \end{array} \parallel \begin{array}{l} r3 = \text{load } y \\ r4 = \text{load } x \end{array} \end{array}$$

(a) Prove that the outcome $r1 = 1, r2 = 0, r3 = 2, r4 = 0$ is **impossible** in the axiomatic TSO model.

(b) Prove that the outcome $r1 = 1, r2 = 0, r3 = 2, r4 = 0$ is **possible** in the axiomatic RA model.

(c) What is the key difference between the TSO rules and the RA rules that makes the outcome possible in one but not the other?

3. Consider the following program:

```
r1 = load x || r2 = load y
store y 1   || store x 1
```

(a) Prove that the outcome $r1 = 1, r2 = 1$ is **impossible** in the axiomatic RA model.

(b) Let the “Relaxed” model be the memory model we get by removing rule 2 from the RA model (so, $hb := po \cup rf$ no longer has to be acyclic). Is the outcome $r1 = 1, r2 = 1$ possible in the Relaxed model? Draw an execution graph that supports your answer.

(c) Consider this variation of the program:

```
r1 = load x || r2 = load y
store y r1  || store x r2
```

What are the possible outcomes of this program under the Relaxed model? (This is a bit of a trick question, so give it your best shot; it will be graded generously.)