0.1 From Examples to Precise Rules

Notations

- T_i denotes thread number i.
- $T_i \equiv T_j$ means both threads have same code.
- w_i^j is the j^{th} event in thread i which is a write.
- r_i^j is the j^{th} event in thread i which is a read.

A few definitions for our use

Definition 1. *Program Order(*po*) Total order between events in the same thread. Respects the execution order between events in the same thread.*

Definition 2. *Symmetric Memory Order (*smo*) A strict partal order between writes in a set of symmetric threads. Consider a set of symmetric threads $T_1 \equiv T_2 \equiv ... \equiv T_n$. Each of these threads have exactly one read event, and multiple write events, all to the same memory, say x.*

Then each write in the above threads are involved in a symmetric order, such that.

$$\forall i \in [0, n-1] \ . \ w_i^j \xrightarrow[smo]{} w_{i+1}^j$$

Where j denotes the j^{th} event in any of the threads, which is a write.

Perhaps should put examples for the above defintion.

Definition 3. *Reads-From (*rf*) Binary relation that links a read to a write from which its value comes. Note that for our purpose, this relation is functional. For example, if a read r_i^j gets its read value from write w_k^l, then we have the relation.*

$$w_k^l \xrightarrow[rf]{} r_i^j$$

Main Rule Using the above setup, our intention is to explore lesser execution graphs leveraging the symmetry that can result due to swapping of thread identities. For this, we enforce a restriction on possible $\xrightarrow[rf]{}$ relations that are to be considered *valid*. A valid $\xrightarrow[rf]{}$ relation is one that respects the following **irreflexivity constraints**.

$$smo; rf; po$$
$$smo; po; rf^{-1}$$

We can add here more as we go about to prove completeness.

Recall examples to show how our analysis through examples satisfy the above irreflexivity constraint.

0.2 Soundness of the rules above

To prove soundness, we first define the following:

Definition 4. *Implied Write Order(*iwo*) Binary relation between any two* distinct *writes, derived through the following two sequential conposition:*

$$w_i^j; po; rf^{-1}; w_k^j$$
$$w_i^j; rf; po; w_k^j$$

Property 1. *Simplified irreflexivity rule*
The irreflexivity constraint rule is equivalent to the following irreflexivity condition

$$smo; iwo \tag{1}$$

Proof. Expanding for implied write order as per the definition, gives us the following two sequential compositions.

$$smo; w_i^j; po; rf^{-1}; w_k^j \tag{2}$$
$$smo; w_i^j; rf; po; w_k^j \tag{3}$$

From the definiton of symmetric order, the above can be simplified to

$$smo; po; rf^{-1} \tag{4}$$
$$smo; rf; po \tag{5}$$

Hence, proving our property. □

Property 2. *No write order is implied when a read reads from its own thread's write*

Proof. If the read is from its own thread's write, then we can infer that $i = k$ in both the sequential compositions. Hence

$$w_i^j; rf; po; w_i^j$$
$$w_i^j; po; rf^{-1}; w_i^j$$

which gives us $w_i^j \xrightarrow[iwo]{} w_i^j$. Since implied write orders are only between distinct writes, the property is proven. □

Property 3. *Implied write orders between two symmetric threads are reversed when they are swapped.*

Proof. Considering first sequential composition, i.e. $w_i^j; rf; po; w_k^j$, expanding gives us the following binary relations involved:

$$w_i^j \xrightarrow[rf]{} r_k \tag{6}$$
$$r_k \xrightarrow[po]{} w_k^j \tag{7}$$

Swapping thread identities involves swapping the indices i and k for each event, thus giving us

$$w_k^j \xrightarrow[rf]{} r_i \tag{8}$$

$$r_i \xrightarrow[po]{} w_i^j \tag{9}$$

Through sequential composition of the above, we get $w_k^j; rf; po; w_i^j$, which by definition is $w_k^j \xrightarrow[iwo]{} w_i^j$.

The argument is symmetric for the second sequential composition. □

Property 4. *There are at most two implied write orders between writes of two threads.*

Proof. Consider two threads T_i and T_k. Suppose we have one implied write order between one of their writes, i.e.

$$w_i^j \xrightarrow[iwo]{} w_k^j. \tag{10}$$

Expanding as per the first sequential composition gives us

$$w_i^j; po; rf^{-1}; w_k^j \tag{11}$$

which also indicates a $\xrightarrow[rf]{}$ with T_i's read and that the writes involved in the composition are above the respective reads.

Now suppose we have an implied write order between another set of writes, i.e.

$$w_i^l \xrightarrow[iwo]{} w_k^l. \tag{12}$$

Expanding as per the first sequential composition of implied write order is not possible as $\xrightarrow[rf]{}$ is functional. Hence, using the second we have

$$w_i^j; rf; po; w_k^j \tag{13}$$

which also indicates a $\xrightarrow[rf]{}$ with T_i's read that the writes involved in the composition are below the respective reads.

Since both reads are now involved in a $\xrightarrow[rf]{}$ relation, and since this relation is functional, we cannot have any more implied write orders between T_i and T_j, thus verifying our property.

Better written in contrast to previous argument. However, is it necessary to show by contradiction? □

Property 5. *Implied write ordes between two threads are either all compliant with $stck_{smo}$ or they are all not*

Proof. If each read reads from its own write, we have no implied write order established, thus maintaining the property.

For cases where implied write orders are established, without loss of generality, let us consider one between writes above the read are compliant with $stck_{smo}$:

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge \ w_i^j \xrightarrow[smo]{} w_k^j \tag{14}$$

The other set of implied write order, if established can only be between writes below the read. Suppose we have such an order but not compliant with $stck_{smo}$:

$$w_k^l \xrightarrow[iwo]{} w_i^l \ \wedge \ w_i^l \xrightarrow[smo]{} w_k^l \tag{15}$$

Upon expanding using the second sequential composition (because writes are below the read), we get

$$w_k^l; rf; po; w_i^l \tag{16}$$

But this implies another $\xrightarrow[rf]{}$ relation with T_i's read, which violates the functional property of it. Hence we can only have an implied write order compliant with $stck_{smo}$.

$$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_i^l \xrightarrow[smo]{} w_k^l \tag{17}$$

Not sure if we need to show that the compliant relation also holds as it only brings an rf realtion with T_k's read, which wanst established before.

The opposite case would make both the implied write orders requiring to not be compliant, thus by symmetry completing our proof. □

Property 6. *Implied write orders are acyclic*

Proof. Suppose a cycle exists. Then without loss of generality, we can consider the cycle composed of 3 writes.

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge w_k^j \xrightarrow[iwo]{} w_l^j \ \wedge w_l^j \xrightarrow[iwo]{} w_i^j \tag{18}$$

If these writes are above the read, then we have the following relations that result in the above cycle.

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf^{-1}]{} w_k^j \tag{19}$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf^{-1}]{} w_l^j \tag{20}$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf^{-1}]{} w_i^j \tag{21}$$

The above relations form a cycle thus violating $po \cup rf^{-1}$ acyclic rule for coherence and hence violating coherence.

If these writes are below the read, then we have the following relations that result in the above cycle.

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf]{} w_k^j \tag{22}$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf]{} w_l^j \tag{23}$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf]{} w_i^j \tag{24}$$

The above relations form a cycle thus violating $po \cup rf$ acyclic rule for coherence and hence violating coherence.

Because both cases violate coherence, we conclude that $\xrightarrow[iwo]{}$ must be acylic.

A bit better written compared to the previous proofs □

Case for two threads

- For every implied write order against symmetric write order, swapping the thread identities reverses the implied write order, thus maintaining the irreflexivity constraint.
- Because all implied write orders between threads are either compliant or not, swapping thread identities does indeed give us an execution compliant with our irreflexivity constraints.
- Hence, for every execution not covered by our rules, there is a symmetric one covered.

Case for three or more threads Method 1:

- I see it as a simple case of bubble sort. Not sure how to show it though.
- Construct a total order for each set of equal writes using implied write orders and the symmetric memory order.
- Then the entire process of constructing a symmetric execution that respects our constraint is just a sorting problem.
- Note that write orders not implied can be freely set by us to respect symmetric memory order.
- How do we show this?

Method 1: For writes above and below (The general case of soundness)

- Proof by contradiction
- Suppose all write orders above reads are sorted as per our need.
- Then for writes below, firstly, if an implied write order is not respected, then either the writes above also not respected or it is a free order.
- If it is not respected, this contradicts our assumption.
- If it is free, then we can freely swap the two threads, fixing the implied wrtie order of the write below the read.
- The above writes will still be in order, because the above write order between those two threads swapped was free, and smo is total w.r.t the equal writes of each symmetric thread.

Method 1: The base case of soundness (one set of equal writes) Part1: Show that collection of $\overrightarrow{iwo}$ w.r.t. one set of equal writes do not form a cycle

- If a cycle exists, we can consider the base case when three writes are involved.
- Showing for three writes that cycle cannot exists extends to the more general case too.
- Two cases :
  - All writes above the read
  - All writes below the read
- For the writes above if we have $w^i_j \overrightarrow{iwo} w^i_k$, then T_j's read reads from T_k's write w^i_k.
- If the three writes above read form a cycle, then this case violates coherence. ($po \cup rf^{-1}$ acyclic.)
- Similarly for the case of writes below the read, if they form a cycle, it is a case violating coherence. ($po \cup rf$ acyclic.)

The above two acyclic parts of coherence is only for events inovlved in one memory operation. The respective generalized forms would be restrictions of Load Buffering or Out of Thin Air values, which is not of our concern when we consider just writes and read to same memory.

Part2: The case for only one set of equal writes

- Show that once an implied write order between two writes is fixed, it will never be un-fixed again.
- To show this, we can keep the implied write order transitive.
- Go by assuming an order to exist between two writes which is correct.
- Consider another implied order between one of these wrties and another outside which is not correct.