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.*

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.*

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

Name to be decided We 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.

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.

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**.

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

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

Place 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:*

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

When the implied write order and symmetric memory order form a cycle, that execution for us is considered invalid. This means the irreflexivity constraint can be also put as :

$$smo; iwo$$

Some observations:

- No write order is implied when a read reads from its own thread's write.
- Implied write orders between two symmetric threads are reversed when they are swapped.
- Either all implied write orders between two threads are compliant with symmetric order, or they are all not. (do an example based analysis, show that it would otherwise violate coherence).

No write order is implied when a read reads from its own thread's write If the read is from its own thread's write, then we can infer that $i = k$ in both the sequential compositions. Hence

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

Since implied write orders only concern with distinct writes, $w_i^j \xrightarrow[iwo]{} w_i^j$ is not of our concern. (irreflexive)

Implied write orders between two symmetric threads are reversed when they are swapped. Considering first sequential composition, i.e. $w_i^j; rf; po; w_k^j$, elaboration gives us the following relations involved:

$$w_i^j \xrightarrow[rf]{} r_k \tag{1}$$
$$r_k \xrightarrow[po]{} w_k^j \tag{2}$$

Swapping thread identities meaning swapping the indics i and k, thus giving us

$$w_k^j \xrightarrow[rf]{} r_i \tag{3}$$
$$r_i \xrightarrow[po]{} w_i^j \tag{4}$$

The above through sequential composition, gives us $w_k^j; rf; po; w_i^j$, which is $w_k^j \xrightarrow[iwo]{} w_i^j$, which is the exact reverse of the original configuration before swapping.

The argument is in the same lines for the second sequential composition.

There are at most two implied write orders between writes of two threads Consider two threads T_i and T_k. Suppose we have one implied write order between one of their writes, i.e. $w_i^j \xrightarrow[iwo]{} w_k^j$. Now if the writes are above the read in program order, then this can be justified by $w_i^j; po; rf^{-1}; w_k^j$, thus involving T_i's read in a $\xrightarrow[rf]{}$ relation. If the writes are below, this can be justified by $w_i^j; rf; po; w_k^j$, involvuing T_k's read in a $\xrightarrow[rf]{}$ relation.

Without loss of generality, let us consider that we have an implied write order between writes above the read. Because $\xrightarrow[rf]{}$ is funcitonal, we cannot have another implied write order between another set of writes above the read such that $w_i^l \xrightarrow[iwo]{} w_k^l$. If we however, consider $w_k^l \xrightarrow[iwo]{} w_i^l$ between the writes above, then it will involve T_k's read in a $\xrightarrow[rf]{}$ relation. However, this violates coherence as the relations established so far violates the coherence rule of $po \cup rf^{-1}$ acyclic. The following relations being involved:

$$w_i^j \xrightarrow[po]{} r_i; \quad r_i \xrightarrow[rf]{} w_k^j; \tag{5}$$

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

Put a figure here to better understand

If we however, consider an implied write order between writes below the read, it can be justified by $w_i^j; rf; po; w_k^j$, involvuing T_k's read in a $\xrightarrow[rf]{}$ relation, without violating Coherence.

Because both reads of T_i and T_k have been involved in a $\xrightarrow[rf]{}$ relation, which is functional by definition, we cannot have more than two implied write orders between two threads. '

Not satisfied with the formal argument because it is too verbal. Think about how to reduce text.

Implied write ordes between two threads are either all compliant with $stck_{smo}$ or they are all not Because we can have at most two implied write orders between two threads and that one of them is above read and one below, we also note that both sequential compositions of implied write orders are used (one for each set of writes).

Without loss of generality, let us consider that the writes above the read are compliant with $stck_{smo}$.

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge \ w_i^j \xrightarrow[smo]{} w_k^j \tag{7}$$

The other set of implied write order can only be between writes below the read. And because T_i's read is already in a $\xrightarrow[rf]{}$ relation, only T_j's read can be used. These two conditions can only be satisfied if $w_i^l; rf; po; w_k^l$, hence giving us seomthing compliant once again with $\xrightarrow[smo]{}$.

$$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_i^l \xrightarrow[smo]{} w_k^l \tag{8}$$

The exact opposite case would make both the implied write orders not compliant, thus completing our proof.

Again, too verbal and not concise enough. Is it just latex laziness?

Implied write orders are acyclic Suppose a cycle exists. Then without loss of generality, we can consider the cycle composed of 3 writes.

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge w_k^j \xrightarrow[iwo]{} w_l^j \ \wedge w_l^j \xrightarrow[iwo]{} w_i^j \tag{9}$$

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

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf^{-1}]{} w_k^j \tag{10}$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf^{-1}]{} w_l^j \tag{11}$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf^{-1}]{} w_i^j \tag{12}$$

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 relations.

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf]{} w_k^j \tag{13}$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf]{} w_l^j \tag{14}$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf]{} w_i^j \tag{15}$$

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 infer that $\xrightarrow[iwo]{}$ is 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 $w_j^i \overrightarrow{iwo} w_k^i$, then T_j's read reads from T_k's write w_k^i.
- 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.