August 29th: One Page Informal Description of Manson/Pugh model

There is a happens-before relation \( \text{hb} \rightarrow \) defined on actions if \( i \) is before \( j \) in program order, if \( i \) is an unlock or volatile write and \( j \) is a matching lock or volatile read that comes after it in the total order over synchronization actions, or if \( i \text{ hb} \rightarrow k \text{ hb} \rightarrow j \) for some \( k \).

A read \( r \) is allowed to see a write \( w \) to the same variable \( v \) if \( r \) does not happen-before \( w \) and if there is no other write \( w' \) to \( v \) such that \( w \text{ hb} \rightarrow w' \text{ hb} \rightarrow r \). An execution that has only allowed reads and respects intra-thread semantics (see Appendix B) is a happens-before consistent execution, or hb-consistent for short.

For every execution, there is a total order over actions, consistent with the synchronization order, called the causal order. Any read action must see a write that occurs earlier in the causal order. An action \( x \) is prescient if there exists an action \( y \) that occurs after \( x \) in the causal order such that \( y \text{ hb} \rightarrow x \).

Each prescient action \( x \) in an execution \( E \) must be justified. Let \( \alpha \) be the sequence of actions that precedes \( x \) in the causal order of \( E \). Let \( J \) be the set of all non-prohibited hb-consistent executions whose causal order consists of \( \alpha \) followed by non-prescient actions (see Appendix C for an algorithm to generate \( J \)). To prove \( x \) is justified, we need to show that for each \( E' \) in \( J \):

- an action \( x' \), congruent to \( x \), occurs in \( E' \) (either \( x' \) and \( x \) are the same action, or they are both reads of the same variable, and it would be hb-consistent for \( x' \) to see the write seen by \( x \)), and

- if \( x \) is a write, then for each read action \( y \in E' \) such that \( y \) reads the same variable as \( x' \) and \( y \text{ hb} \rightarrow x' \), we need to show \( y \in \alpha \).

Justification may involve the use of prohibited executions. Prohibited executions are defined by a set of prohibited causal order prefixes \( P \). Given \( P \), an execution \( E \) is prohibited if the causal order for \( E \) starts with a prefix in \( P \) (typically, \( P \) is empty and no executions are prohibited).

A set of prohibited prefixes must be valid. To show that a set of prohibited prefixes is valid, we must show that:

- For each prefix \( \alpha x \in P \), there exists some non-prohibited execution \( E \) with a causal order \( \alpha \beta \) such that \( \beta \) contains no prescient actions.

- Consider any execution \( E \) with causal order \( \alpha xy \beta \) where \( x \) and \( y \) are not conflicting actions and are not both synchronization actions. Also consider the execution \( E' \) with causal order \( \alpha yx \beta \). Execution \( E \) by be forbidden by \( P \) if and only if \( E' \) is forbidden.

The interpretation of having \( \alpha x \) as a prohibited prefix is that whenever an execution starts with the causal order prefix \( \alpha \), the action \( x \) is prohibited as the next action in the causal order.

Given these definitions, an hb-consistent execution \( E \) is legal if and only if there exists a set of prohibited prefixes \( P \) such that \( E \) is not prohibited by \( P \) and using \( P \) as the prohibited prefixes, all of the prescient actions in \( E \) are justified.
Appendix

These appendices include clarifications that have been requested.

A Differences with Old Model

Here is a brief rundown on the differences between the new model and the model in the community review draft.

- Consistency is now called \( hb\text{-consistency} \).
- Previously, we allowed a prescient read action to see a write that occurs later in the causal order.
  Now all reads must see writes that occur earlier in the causal order.
- A write \( w \) cannot occur presciently if in the justifying execution there is a conflicting read \( r \) such that \( r \xrightarrow{hb} w \).
- Prohibited sets are defined in a slightly different way. In particular, they are global, so that in order to justify an action \( x \) in an execution \( E \), you may not prohibit \( E \).

B Intra-thread Semantics

Given an execution where each read sees a write that it is allowed to see by the happens-before constraint, we verify that the execution respects intra-thread semantics as follows. For each thread \( t \), we go through the actions of that thread in program order. For each non-read action \( x \), we verify that the behavior of that action is what would follow from the previous actions in that thread according to the JLS/JVMS. For a read action, we only verify that the variable read is the one that is determined by the previous actions in the thread according to the JLS; the value seen by the read is determined by the memory model.

C Generating Non-prescient Extensions

Say we have a program \( P \), and a partial causal order \( \alpha \). We can compute the set of all non-prescient extensions to \( \alpha \) as follows.

- Let \( S \) be a set of partial and complete causal orders, initialized to be the singleton set containing \( \alpha \).
- Let \( W \) be a worklist of causal orders to be explored, initialized to \( S \).
- While \( W \) is non-empty, choose and remove a causal order \( \beta \) from \( W \)
– For each thread \( t \) in \( P \), select the first statement in program order whose execution is not in \( \beta \).
  * If that statement is not a read, then evaluate that statement in the thread-local context of \( \beta \), generating action \( x \), and add \( \beta x \) to both \( S \) and \( W \).
  * If that statement is a read, determine, in the thread-local context of \( \beta \), which variable \( v \) will be read. For each write \( w \in \beta \) of \( v \) that could be seen by the read, generate the action \( r \) corresponding to that read seeing \( w \), and add \( \beta r \) to both \( S \) and \( W \).

• When \( W \) is empty, the complete causal orders in \( S \) corresponding to hb-consistent executions are the non-prescient extensions to \( \alpha \).