An execution trace consists of a set of actions, a happens-before ordering over those actions that is the partial order derived from those actions, and a causal order, which is a total order over all of the actions in the trace.

We use \( \langle S, \rightarrow^{hb}, \rightarrow \rangle \) to represent an execution trace \( E \), where

- \( S \) is a set of actions,
- \( \rightarrow^{hb} \) is a partial order over the actions in \( S \), and
- \( \rightarrow \) is the causal order: an ordered list of all the actions in \( S \).

An execution trace is consistent if the actions performed are consistent with the intra-thread semantics of the program and each read observes the value of a write that it is allowed to observe by the happens before ordering.

A consistent execution trace \( E = \langle S, \rightarrow^{hb}, \rightarrow \rangle \) is also causal, and therefore valid, if and only if there exists a set of prohibited executions such that each prescient action \( x \) in \( S \) is justified. (Feel free to ignore prohibited executions on first reading; they only come into play on certain corner cases).

An action \( x \) in a trace \( \langle S, \rightarrow^{hb}, \rightarrow \rangle \) is prescient if and only if there exists an action \( y \) that occurs after \( x \) in the causal order \( \rightarrow \) such that either \( y \rightarrow^{hb} x \), or \( x \) is a read, \( y \) is a write, and \( x \) observe \( y \).

All prescient actions must be justified. To justify a prescient action \( x \) in trace \( E \), we need to show that the actions before \( x \) in the causal order guarantee that \( x \) will be allowed:

- Let \( \alpha \) be the prefix of \( x \) in the causal order for \( E \)
- Define, \( J \), the justification for \( x \), as

\[
J = \{ E' = \langle S', \rightarrow^{hb'}, \alpha' \beta' \rangle \mid E' \text{ is consistent and not prohibited} \\
\wedge \text{length}(\alpha) = \text{length}(\alpha') \\
\wedge \beta' \text{ does not contain prescient actions} \\
\wedge \alpha \preceq \alpha' \beta' \}
\]

- For \( x \) to be justified, \( J \) must be non-empty and for each \( E' = \langle S', \rightarrow^{hb'}, \alpha' \beta' \rangle \) in \( J \), there must exist an action \( x' \) in \( \beta' \) such that \( x' \mapsto x \).
Prohibited Alternative Executions

For the purposes of showing that a prescient action \( x \) is justified, a set of behaviors that are not possible on a particular implementation of a JVM may be specified. This, in turn, allows other actions to be guaranteed and performed presciently, allowing for new behaviors.

This is handled by specifying a list of alternative executions \([AE_1, AE_2, \ldots, AE_n]\), each alternative execution \( AE_i \) consisting of a prohibited execution \( E \) and a preferred alternative execution \( E' \):

\[
AE_i = \langle E_i = \langle S_i, \text{hb}_i, \alpha_i, r_i, \beta_i \rangle, E'_i = \langle S'_i, \text{hb}'_i, \alpha'_i, r'_i, \beta'_i \rangle \rangle
\]

The intuition here is that execution \( AE_i \) would not occur, because behavior \( AE'_i \) would occur instead. Define \( \text{valid}_0 \) be the set of executions that are causal and consistent without any use of alternative executions. Define \( \text{valid}_k \) to be the set of executions shown to be causal by prohibiting the executions \( \{E_1, E_2, \ldots, E_k\} \).

For a list of alternative executions to be usable, for all \( k \),

- \( E'_k \) must be in \( \text{valid}_{k-1} \),
- \( \alpha_k \preceq \alpha'_k \),
- \( r_k \mapsto r'_k \), and
- \( r_k \) must observe a different write than \( r'_k \).