A Appendix

A.1 JavaScript Well-formedness

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{range}_r(E : \mathit{event}) & \triangleq & [\ E.\mathsf{index} \ldots E.\mathsf{index} + |E.\mathsf{reads}|\ ) & \mathsf{read}(E : \mathit{event}) & \triangleq & (E.\mathsf{reads} \neq [\,]) \\
\mathsf{range}_w(E : \mathit{event}) & \triangleq & [\ E.\mathsf{index} \ldots E.\mathsf{index} + |E.\mathsf{writes}|\ ) & \mathsf{write}(E : \mathit{event}) & \triangleq & (E.\mathsf{writes} \neq [\,]) \\
\mathsf{range}(E : \mathit{event}) & \triangleq & \mathsf{range}_r(E) \cup \mathsf{range}_w(E) & \mathsf{rmw}(E : \mathit{event}) & \triangleq & \mathsf{read}(E) \wedge \mathsf{write}(E) \\
\mathsf{overlap}(E_1, E_2 : \mathit{event}) & \triangleq & E_1.\mathsf{block} = E_2.\mathsf{block} \wedge & \mathsf{read}(Es : \mathit{set\ event}) & \triangleq & \{E \mid E \in Es \wedge \mathsf{read}(E)\} \\
 & & \quad \mathsf{range}(E_1) \cap \mathsf{range}(E_2) \neq \emptyset & \mathsf{write}(Es : \mathit{set\ event}) & \triangleq & \{E \mid E \in Es \wedge \mathsf{write}(E)\}
\end{array}$$

$$\begin{array}{l}
\mathsf{well\text{-}formed}(k : \mathit{nat}, E_w : \mathit{event}, E_r : \mathit{event}) \triangleq \\
\quad k \in \mathsf{range}_w(E_w) \wedge k \in \mathsf{range}_r(E_r) \wedge E_w.\mathsf{block} = E_r.\mathsf{block} \wedge \\
\quad (E_w.\mathsf{writes})!(k - E_w.\mathsf{index}) = (E_r.\mathsf{reads})!(k - E_r.\mathsf{index}) \wedge E_w \neq E_r
\end{array}$$

$$\begin{array}{l}
\mathsf{well\text{-}formed}(\mathit{rbf} : \mathit{set}\ (\mathit{nat} \times \mathit{event} \times \mathit{event}), Es : \mathit{set\ event}) \triangleq \\
\quad (\ \forall E_r \in Es, k \in \mathsf{range}_r(E_r).\ \exists! E_w.\ \langle k, E_w, E_r\rangle \in \mathit{rbf}\ ) \wedge \\
\quad (\ \forall E_r \in Es, k \notin \mathsf{range}_r(E_r).\ \nexists E_w.\ \langle k, E_w, E_r\rangle \in \mathit{rbf}\ ) \wedge \\
\quad \forall\ \langle k, E_w, E_r\rangle \in \mathit{rbf}.\ \mathsf{well\text{-}formed}(k, E_w, E_r)
\end{array}$$

$$\begin{array}{l}
\mathsf{well\text{-}formed}(\mathit{CE} : \mathit{candidate_execution}) \triangleq \\
\quad \mathit{CE}.\mathsf{total\text{-}order\ is\ a\ strict\ total\ order\ on}\ \mathit{CE}.\mathsf{evs} \wedge \\
\quad \mathsf{well\text{-}formed}(\mathit{CE}.\mathsf{reads\text{-}byte\text{-}from}, \mathit{CE}.\mathsf{evs})
\end{array}$$

Figure 14. Candidate Execution Well-formedness

A.2 Comparison to Prose JavaScript Specification

The model as presented by the paper is intended to be a faithful rendering in logic of the model as it is described in the JavaScript specification. However, we sometimes factor or name definitions differently for the sake of a simpler presentation or to match standard terminology in the literature. Noticeable differences are highlighted here.

- The prose specification refers to written values as an event's "payload". Read events do not contain a payload field, but their read value is contained in a separate "chosen value record". We simply make both of these fields part of the event as explicit read/write fields.
- The prose specification uses the term memory-order to refer to what we call total-order, but this invites confusion with the C/C++11 memory model: there a "memory order" is what JavaScript calls a "consistency mode", and "mo" refers to "modification order", a per-location-total order on writes [9].
- In addition, the prose specification existentially quantifies **tot** as part of its validity conditions. We make this a field of the candidate execution.
- The prose specification does not make a distinction between well-formedness conditions (which are purely invariants of the thread-local semantics), and validity conditions, which are semantic conditions imposed by the axiomatic memory model. We separate well-formedness from validity in our rendering of the model because in later sections we explicitly want to reason about executions which are well-formed but nevertheless not valid.
- In the prose specification our Happens-Before Consistency (2-3) conditions are collectively named the "Coherent Reads" condition.
- In several cases, the prose specification defines predicates using a pseudocode algorithm. For example, the prose specification's "Coherent Reads" condition is defined as an explicit iteration over the set of events, accumulating a boolean result, instead of as a more abstract universal quantification.
  Our Happens-Before Consistency (1) condition is instead given as a component of Sequentially Consistent Atomics.
- The prose specification describes an additional condition that all atomic writes must have a finite prefix in total-order. This condition is vacuously true in real candidate executions, since total-order must be well-founded (every thread must have a first event in sequenced-before-order). This condition is occasionally included in formal relaxed memory models as a well-formedness condition that is implied by the thread-local semantics [9]. However, we never need to use it, so we simply elide it here.

A.3 Comparison to Alloy/Coq models

As previously detailed, our proofs assume that all initialization happens "before time". The Coq model undergoes two minor refactorings in order to better interface with the existing IMM models. First, validity conditions have individual edges reversed so that they can be expressed as equivalent acyclicity constraints. Second, RMW events are equivalently represented as two events which are always adjacent in **sb** and **tot**. Both of these transformations are analogous to the transformations made to other models (such as C/C++11) in previous works in order to simplify formal proofs [28, 35]

B ARMv8 model

$$
\begin{array}{rcll}
\mathit{mode} & ::= & \textbf{ReadAcquire} & \\
 & | & \textbf{ReadWeakAcquire} & \\
 & | & \textbf{WriteRelease} & \\
 & | & \textbf{DMB.SY} & \\
 & | & \textbf{DMB.LD} & \\
 & | & \textbf{DMB.ST} & \\
 & | & \textbf{ISB} & \\
\mathit{addr} & ::= & \alpha \ldots \text{ an infinite set of abstract names} & \\
\mathit{event} & ::= & \{\, \text{ord} & : \mathit{mode} \\
 & & \text{index} & : \mathit{nat} \\
 & & \text{reads} & : \mathit{list\ byte} \\
 & & \text{writes} & : \mathit{list\ byte} \\
\mathit{candidate\_execution} & ::= & \{\, \text{evs} & : \mathit{set\ event} \\
 & & \text{IW} & : \mathit{event} \\
 & & \text{program-order} \mathbin{/\!/} \textbf{po} & : \mathit{set}\ (\mathit{nat} \times \mathit{event} \times \mathit{event}) \\
 & & \text{reads-byte-from} \mathbin{/\!/} \textbf{rbf} & : \mathit{set}\ (\mathit{nat} \times \mathit{event} \times \mathit{event}) \\
 & & \text{bytewise-coherence} \mathbin{/\!/} \textbf{cob} & : \mathit{set}\ (\mathit{nat} \times \mathit{event} \times \mathit{event}) \\
 & & \text{address-dependency-pre} \mathbin{/\!/} \textbf{addrP} & : \mathit{set}\ (\mathit{event} \times \mathit{event}) \\
 & & \text{data-dependency-pre} \mathbin{/\!/} \textbf{dataP} & : \mathit{set}\ (\mathit{event} \times \mathit{event}) \\
 & & \text{control-dependency-pre} \mathbin{/\!/} \textbf{ctrlP} & : \mathit{set}\ (\mathit{event} \times \mathit{event}) \\
 & & \text{read-modify-write} \mathbin{/\!/} \textbf{rmw} & : \mathit{set}\ (\mathit{event} \times \mathit{event})\ \}
\end{array}
$$

Following herd [6], we define relational composition ";" the identity relation "[A]" on a set A:

$$
\begin{aligned}
R; S &\triangleq \{\langle A, B\rangle \mid \exists C.\ \langle A, C\rangle \in R \wedge \langle C, B\rangle \in S\} \\
[E] &\triangleq \{\langle E, E\rangle \mid E \in A\}
\end{aligned}
$$

Then, $[A]; R; [B]$ is the relation R restricted to elements from A on the left and elements from B from the right:

$$
[A]; R; [B] = \{\langle a, b\rangle \mid \langle a, b\rangle \in R \wedge a \in A \wedge b \in B\}
$$

Derived auxiliary relations *(with respect to a candidate execution)*

$$
\begin{aligned}
\text{same-thread} \mathbin{/\!/} \textbf{sthd} &\triangleq \{\langle A, B\rangle \mid \langle A, B\rangle \in \textbf{po} \vee \langle B, A\rangle \in \textbf{po}\} \\
\text{reads-from} \mathbin{/\!/} \textbf{rf} &\triangleq \{\langle A, B\rangle \mid \exists k.\ \langle k, A, B\rangle \in \text{reads-byte-from}\} \\
\text{coherence} \mathbin{/\!/} \textbf{co} &\triangleq \{\langle A, B\rangle \mid \exists k.\ \langle k, A, B\rangle \in \text{bytewise-coherence}\}^{+} \\
\text{reads-from-internal} \mathbin{/\!/} \textbf{rfi} &\triangleq \text{reads-from} \cap \text{same-thread} \\
\text{reads-from-external} \mathbin{/\!/} \textbf{rfe} &\triangleq \text{reads-from} \setminus \text{reads-from-internal} \\
\text{coherence-internal} \mathbin{/\!/} \textbf{coi} &\triangleq \text{coherence} \cap \text{same-thread} \\
\text{coherence-external} \mathbin{/\!/} \textbf{coe} &\triangleq \text{coherence} \setminus \text{coherence-internal} \\
\text{bytewise-from-reads} \mathbin{/\!/} \textbf{frb} &\triangleq \{\langle k, A, B\rangle \mid \exists C.\ \langle k, C, A\rangle \in \text{reads-byte-from} \wedge \langle k, C, B\rangle \in \text{bytewise-coherence}\}^{+} \\
\text{from-reads} \mathbin{/\!/} \textbf{fr} &\triangleq \{\langle A, B\rangle \mid \exists k.\ \langle k, A, B\rangle \in \text{bytewise-from-reads}\}^{+} \\
\text{from-reads-internal} \mathbin{/\!/} \textbf{fri} &\triangleq \text{from-reads} \cap \text{same-thread} \\
\text{from-reads-external} \mathbin{/\!/} \textbf{fre} &\triangleq \text{from-reads} \setminus \text{from-reads-internal} \\
\text{program-order-same-byte-location} \mathbin{/\!/} \textbf{polocb} &\triangleq \{\langle k, A, B\rangle \mid \langle A, B\rangle \in \textbf{po} \wedge k \in \text{range}(A) \wedge k \in \text{range}(B)\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{R} &\triangleq \{E \mid E.\mathsf{reads} \neq []\} \\
\mathsf{W} &\triangleq \{E \mid E.\mathsf{writes} \neq []\} \\
\mathsf{A} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{ReadAcquire}\} \\
\mathsf{Q} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{ReadWeakAcquire}\} \\
Ł &\triangleq \{E \mid E.\mathsf{ord} = \textbf{WriteRelease}\} \\
\mathsf{DMB.SY} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{DMB.SY}\} \\
\mathsf{DMB.LD} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{DMB.LD}\} \\
\mathsf{DMB.ST} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{DMB.ST}\} \\
\mathsf{ISB} &\triangleq \{E \mid E.\mathsf{ord} = \textbf{ISB}\} \\
\mathsf{B} &\triangleq \mathsf{DMB.SY} \cup \mathsf{DMB.LD} \cup \mathsf{DMB.ST} \cup \mathsf{ISB}
\end{aligned}
$$

$$
\begin{aligned}
\text{address-dependency } // \textbf{addr} &\triangleq [R]; \textbf{addrP} \\
\text{data-dependency } // \textbf{data} &\triangleq [R]; \textbf{dataP} \\
\text{control-dependency } // \textbf{ctrl} &\triangleq [R]; \textbf{ctrlP}
\end{aligned}
$$

$$
\begin{aligned}
\text{coherence-after } // \textbf{ca} &\triangleq \textbf{co} \cup \textbf{fr} \\
\text{observed } // \textbf{obs} &\triangleq \textbf{rfe} \cup \textbf{fre} \cup \textbf{coe}
\end{aligned}
$$

$$
\begin{aligned}
\text{dependency-ordered-before } // \textbf{dob} \triangleq\ & \textbf{addr} \cup \\
& \textbf{data} \cup \\
& (\textbf{ctrl}; [\mathsf{W}]) \cup \\
& ((\textbf{ctrl} \cup \textbf{addr}; \textbf{po}); [\mathsf{ISB}]; \textbf{po}; [\mathsf{R}]) \cup \\
& (\textbf{addr}; \textbf{po}; [\mathsf{W}]) \cup \\
& ((\textbf{ctrl} \cup \textbf{data}); \textbf{coi})) \cup \\
& ((\textbf{addr} \cup \textbf{data}); \textbf{rfi})) \\
\text{atomic-ordered-before } // \textbf{aob} \triangleq\ & \textbf{rmw} \cup (\textbf{rmw}; [R]; \textbf{rfi}; [\mathsf{A} \cup \mathsf{Q}]) \\
\text{barrier-ordered-before } // \textbf{bob} \triangleq\ & \textbf{po}; [\mathsf{DMB.SY}]; \textbf{po} \cup \\
& ([\mathsf{L}]; \textbf{po}; [\mathsf{A}]) \cup \\
& ([\mathsf{R}]; \textbf{po}; [\mathsf{DMB.LD}]; \textbf{po}) \cup \\
& ([\mathsf{A} \cup \mathsf{Q}]; \textbf{po}) \cup \\
& ([\mathsf{W}]; \textbf{po}; [\mathsf{DMB.ST}]; \textbf{po}; [\mathsf{W}]) \cup \\
& (\textbf{po}; [\mathsf{L}]) \cup \\
& (\textbf{po}; [\mathsf{L}]; \textbf{coi}) \cup
\end{aligned}
$$

$$
\text{initial-ordered-before } // \textbf{iob} \triangleq \{(\mathsf{IW}, E) \mid E \in \mathsf{EV} \setminus \mathsf{IW}\}
$$

$$
\text{ordered-before } // \textbf{ob} \triangleq \textbf{obs} \cup \textbf{dob} \cup \textbf{aob} \cup \textbf{bob}
$$

Well-formedness conditions

Let R_k, for some relation $R : \mathit{set}\ (\mathit{nat} \times \mathit{event} \times \mathit{event})$, denote the relation $\{\langle A, B\rangle \mid \langle k, A, B\rangle \in R\} : \mathit{set}\ (\mathit{event} \times \mathit{event})$.

- The initial write is not a release write:

$$\mathsf{IW} \subseteq W \setminus L$$

- In our fragment, reads and writes are disjoint, and reads, writes, and barriers are disjoint:

$$\mathsf{R} \cap \mathsf{W} \cap \mathsf{DMB.SY} \cap \mathsf{DMB.LD} \cap \mathit{DMBST} \cap \mathit{ISB} = \emptyset$$

- Barriers don't read or write memory:

$$\{\mathsf{range}(E) \mid E \in \mathsf{B}\} = \emptyset$$

- For all locations k, $\mathbf{cob}_k$ is a strict total order on $\{W \mid k \in W.\mathsf{writes}\}$ ...
- ... and only includes writes that write to k:

$$\forall k, W, W'.\ \langle W, W'\rangle \in \mathbf{cob}_k \implies k \in W.\mathsf{writes} \cap W'.\mathsf{writes}$$

- coherence is a strict partial order.
- read-modify-write, address-dependency, data-dependency, and control-dependency are subsets of program-order:

$$\mathbf{rmw} \cup \mathbf{addr} \cup \mathbf{data} \cup \mathbf{ctrl} \subseteq \mathbf{po}$$

Candidate execution validity

Single-copy atomicity
(**rf**; **fr**) irreflexive

Coherence/internal
$\forall k.(\mathbf{polocb}_k \cup \mathbf{frb}_k \cup \mathbf{cob}_k \cup \mathbf{rbf}_k)$ acyclic

External
ob acyclic

Exclusives/Atomic
$\mathbf{rmw} \cap (\mathbf{fre}; \mathbf{coe}) = \emptyset$