Overhauling SC atomics in C11 and OpenCL

Mark Batty, Alastair F. Donaldson and John Wickerson
In short

• The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are
The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are 😕 too weak, 😠 too strong, and 😞 too complex.

In short

- The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are

  😞 too complex,
In short

- The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are
  😞 too complex,
  😞 too weak, and
In short

- The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are:
  - too complex,
  - too weak, and
  - too strong.
In short

• The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are 😞 too complex,

 😞 too weak, and

 😠 too strong.

• We suggest how to fix them 😊.
Outline

- Introduction to the C11 memory model
- Overhauling the rules for SC atomics in C11
- Introduction to the OpenCL memory model
- Overhauling the rules for SC atomics in OpenCL
The C11 memory model

- Non-atomics
- Relaxed atomics
- Acquire/release atomics
- SC atomics
\[ *x = 42; \]
\[
\text{atomic\_store\_explicit}(y, 1, \text{memory\_order\_release});
\]
\[
\text{if (atomic\_load\_explicit}(y, \text{memory\_order\_acquire})) \]
\[
\text{print(*x);}
\]
Consistent executions

• Execution $X$ is consistent iff

satisfies all the consistency axioms.
Consistent executions

- Execution $X$ is **consistent** iff there exists $rf$, $mo$ and $S$ such that $(X, rf, mo, S)$ is well-formed and satisfies all the **consistency axioms**.
Candidate executions

\[
\begin{align*}
a: \ W_{na}(x, 0) & \quad b: \ W_{na}(y, 0) \\
c: \ W(x, 1, RLX) & \quad d: \ R(x, 1, RLX) & \quad f: \ W(x, 2, SC) & \quad h: \ W(y, 1, SC) \\
\quad & \downarrow \text{sb} & \quad & \downarrow \text{sb} & \quad & \downarrow \text{sb} \\
e: \ R(x, 2, RLX) & \quad g: \ R(y, 0, SC) & \quad i: \ R(x, 1, SC)
\end{align*}
\]
Candidate executions

\[
a: W_{na}(x, 0) \quad b: W_{na}(y, 0) \\
c: W(x, 1, RLX) \quad d: R(x, 1, RLX) \quad f: W(x, 2, SC) \quad h: W(y, 1, SC) \\
e: R(x, 2, RLX) \quad g: R(y, 0, SC) \\
i: R(x, 1, SC)
\]
All consistency axioms

\textbf{irr}(hb)
\textbf{irr}(rf^{-1}? ; mo ; rf? ; hb)
\textbf{irr}(rf ; hb)
\textbf{empty}((rf ; [nal]) \setminus \text{vis})
\textbf{irr}(rf \cup (mo ; mo ; rf^{-1}) \cup (mo ; rf))
\textbf{irr}(S ; hb)
\textbf{irr}(S ; Fsb? ; mo ; sbF?)
\textbf{irr}(S ; rf^{-1} ; [SC] ; mo)
\textbf{irr}((S \setminus (mo ; S)) ; rf^{-1} ; hbl ; [W])
\textbf{irr}(S ; Fsb ; rf^{-1} ; mo)
\textbf{irr}(S ; rf^{-1} ; mo ; sbF)
\textbf{irr}(S ; Fsb ; rf^{-1} ; mo ; sbF)
Derived relations

\[ F_{sb} = [F] \cup sb \]

\[ sbF = sb \cup [F] \]

\[ rs' = \text{thd} \cup (E^2 \cap (R \cap W)) \]

\[ rs = mo \cap rs' \setminus ((mo \setminus rs') \cup mo) \]

\[ sw = ([rel] \cup F_{sb} \cup [W \cap A] \cup rs \cup rf \cup [R \cap A] \cup sbF \cup [acq]) \setminus \text{thd} \]

\[ hb = (sb \cup (I \times I) \cup sw)^+ \]

\[ hbl = hb \cap \text{loc} \]

\[ vis = (W \times R) \cap hbl \setminus (hbl \cup [W] \cup hb) \]
Outline

• Introduction to the C11 memory model
• Overhauling the rules for SC atomics in C11
• Introduction to the OpenCL memory model
• Overhauling the rules for SC atomics in OpenCL
All consistency axioms

\( \text{irr}(\text{hb}) \)
\( \text{irr}(\text{rf}^{-1}? ; \text{mo} ; \text{rf}? ; \text{hb}) \)
\( \text{irr}(\text{rf} ; \text{hb}) \)
\( \text{empty}(((\text{rf} ; [\text{nal}]) \setminus \text{vis}) \)
\( \text{irr}(\text{rf} \cup (\text{mo} ; \text{mo} ; \text{rf}^{-1}) \cup (\text{mo} ; \text{rf})) \)
\( \text{irr}(S ; \text{hb}) \)
\( \text{irr}(S ; \text{Fsb}? ; \text{mo} ; \text{sbF}? ) \)
\( \text{irr}(S ; \text{rf}^{-1} ; [\text{SC}] ; \text{mo}) \)
\( \text{irr}((S \setminus (\text{mo} ; S)) ; \text{rf}^{-1} ; \text{hbl} ; [\text{W}]) \)
\( \text{irr}(S ; \text{Fsb} ; \text{rf}^{-1} ; \text{mo}) \)
\( \text{irr}(S ; \text{rf}^{-1} ; \text{mo} ; \text{sbF}) \)
\( \text{irr}(S ; \text{Fsb} ; \text{rf}^{-1} ; \text{mo} ; \text{sbF}) \)
SC axioms

\texttt{irr}(S ; hb)
\texttt{irr}(S ; Fsb? ; mo ; sbF?)
\texttt{irr}(S ; rf^{-1} ; [SC] ; mo)
\texttt{irr}((S \setminus (mo ; S)) ; rf^{-1} ; hbl ; [W])
\texttt{irr}(S ; Fsb ; rf^{-1} ; mo)
\texttt{irr}(S ; rf^{-1} ; mo ; sbF)
\texttt{irr}(S ; Fsb ; rf^{-1} ; mo ; sbF)
SC axioms

\[
\text{irr}(S; \text{hb}) \\
\text{irr}(S; \text{Fsb?}; \text{mo}; \text{sbF?}) \\
\text{irr}(S; \text{rf}^{-1}; \text{[SC]}; \text{mo}) \\
\text{irr}((S \setminus (\text{mo}; S)); \text{rf}^{-1}; \text{hbl}; \text{[W]}) \\
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}) \\
\text{irr}(S; \text{rf}^{-1}; \text{mo}; \text{sbF}) \\
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}; \text{sbF})
\]
SC axioms

\[
\begin{align*}
\text{irr}(S ; & \quad \text{hb}) \\
\text{irr}(S ; & \quad \text{Fsb? ; mo ; sbF?}) \\
\text{irr}(S ; & \quad \text{rf}^{-1} ; [\text{SC}] ; \text{mo}) \\
\text{irr}((S \setminus (\text{mo} ; S)) ; & \quad \text{rf}^{-1} ; \text{hbl} ; [W]) \\
\text{irr}(S ; & \quad \text{Fsb} ; \text{rf}^{-1} ; \text{mo}) \\
\text{irr}(S ; & \quad \text{rf}^{-1} ; \text{mo} ; \text{sbF}) \\
\text{irr}(S ; & \quad \text{Fsb} ; \text{rf}^{-1} ; \text{mo} ; \text{sbF})
\end{align*}
\]
SC axioms

\[
\text{irr}(S; \text{hb})
\]

\[
\text{irr}(S; \text{Fsb?}; \text{mo}; \text{sbF?})
\]

\[
\text{irr}(S; \text{rf}^{-1}; [\text{SC}]; \text{mo})
\]

\[
\text{irr}((S \backslash (\text{mo} ; S)); \text{rf}^{-1}; \text{hbl}; [\text{W}])
\]

\[
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo})
\]

\[
\text{irr}(S; \text{rf}^{-1}; \text{mo}; \text{sbF})
\]

\[
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}; \text{sbF})
\]
SC axioms

\[
\begin{align*}
\text{irr}(S; & \text{ hb}) \\
\text{irr}(S; & \text{ Fsb?; mo; sbF?}) \\
\text{irr}(S; & \text{ rf}^{-1}; [SC]; \text{ mo}) \\
\text{irr}((S \setminus (\text{ mo} \cup S)); & \text{ rf}^{-1}; \text{ hbl}; [W]) \\
\text{irr}(S; & \text{ Fsb; rf}^{-1}; \text{ mo}) \\
\text{irr}(S; & \text{ rf}^{-1}; \text{ mo}; \text{ sbF}) \\
\text{irr}(S; & \text{ Fsb; rf}^{-1}; \text{ mo}; \text{ sbF})
\end{align*}
\]
Candidate executions

\[ a: W_{na}(x, 0) \]
\[ b: W_{na}(y, 0) \]
\[ c: W(x, 1, \text{RLX}) \]
\[ d: R(x, 1, \text{RLX}) \]
\[ e: R(x, 2, \text{RLX}) \]
\[ f: W(x, 2, \text{SC}) \]
\[ g: R(y, 0, \text{SC}) \]
\[ h: W(y, 1, \text{SC}) \]
\[ i: R(x, 1, \text{SC}) \]
SC axioms

\[
\begin{align*}
\text{irr}(S ; & \quad \text{hb}) \\
\text{irr}(S ; & \quad Fsb? ; \text{mo} ; \text{sbF?}) \\
\text{irr}(S ; & \quad rf^{-1} ; [SC] ; \text{mo}) \\
\text{irr}((S \setminus (\text{mo} \cup S)) ; & \quad rf^{-1} ; \text{hbl} ; [W]) \\
\text{irr}(S ; & \quad Fsb ; rf^{-1} ; \text{mo}) \\
\text{irr}(S ; & \quad rf^{-1} ; \text{mo} ; \text{sbF}) \\
\text{irr}(S ; & \quad Fsb ; rf^{-1} ; \text{mo} ; \text{sbF})
\end{align*}
\]
SC axioms

\[
\text{irr}(S \ ; \ \text{hb})
\]
\[
\text{irr}(S \ ; \ \text{Fsb?} \ ; \ \text{mo} \ ; \ \text{sbF?})
\]
\[
\text{irr}(S \ ; \ \text{rf}^{-1} \ ; \ \text{mo})
\]
\[
\text{irr}(S \ ; \ \text{rf}^{-1} \ ; \ \text{hbl} \ ; \ [W])
\]
\[
\text{irr}(S \ ; \ \text{Fsb} \ ; \ \text{rf}^{-1} \ ; \ \text{mo})
\]
\[
\text{irr}(S \ ; \ \text{rf}^{-1} \ ; \ \text{mo} \ ; \ \text{sbF})
\]
\[
\text{irr}(S \ ; \ \text{Fsb} \ ; \ \text{rf}^{-1} \ ; \ \text{mo} \ ; \ \text{sbF})
\]
SC axioms

\text{irr}(S; \text{hb})

\text{irr}(S; \text{Fsb?}; \text{mo}; \text{sbF?})

\text{irr}(S; \text{rf}^{-1}; \text{mo})

\text{irr}(S; \text{rf}^{-1}; \text{hbl}; \text{[W]})

\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo})

\text{irr}(S; \text{rf}^{-1}; \text{mo}; \text{sbF})

\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}; \text{sbF})
SC axioms

\text{irr}(S; \text{hb})
\text{irr}(S; \text{Fsb?}; \text{mo}; \text{sbF?})
\text{irr}(S; \text{rf}^{-1}; \text{mo})
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo})
\text{irr}(S; \text{rf}^{-1}; \text{mo}; \text{sbF})
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}; \text{sbF})
SC axioms

\text{irr}(S; \text{hb})
\text{irr}(S; \text{Fsb}\,?; \text{mo}; \text{sbF}\,?)
\text{irr}(S; \text{rf}^{-1}; \text{mo})
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo})
\text{irr}(S; \text{rf}^{-1}; \text{mo}; \text{sbF})
\text{irr}(S; \text{Fsb}; \text{rf}^{-1}; \text{mo}; \text{sbF})
SC axioms

\[
\text{irr}(S; \text{ hb})
\]

\[
\text{irr}(S; \text{ Fsb?}; \text{ mo}; \text{ sbF?})
\]

\[
\text{irr}(S; \text{ rf}^{-1}; \text{ mo})
\]

\[
\text{irr}(S; \text{ Fsb}; \text{ rf}^{-1}; \text{ mo})
\]

\[
\text{irr}(S; \text{ rf}^{-1}; \text{ mo}; \text{ sbF})
\]

\[
\text{irr}(S; \text{ Fsb}; \text{ rf}^{-1}; \text{ mo}; \text{ sbF})
\]
SC axioms

\texttt{irr}(S ; \text{hb})

\texttt{irr}(S ; \text{Fsb? ; mo ; sbF?})

\texttt{irr}(S ; \text{Fsb? ; rf}^{-1} ; \text{mo ; sbF?})
SC axioms

\[ \text{irr}(S; \text{hb}) \]
\[ \text{irr}(S; Fsb?; \text{mo}; \text{sbF?}) \]
\[ \text{irr}(S; Fsb?; rf^{-1}; \text{mo}; \text{sbF?}) \]
SC axioms

\textbf{irr}(S ; hb)
\textbf{irr}(S ; Fsb? ; mo ; sbF?)
\textbf{irr}(S ; Fsb? ; rf^{-1} ; mo ; sbF?)
SC axioms

\textbf{irr}(S ; Fsb? ; hb ; sbF?)
\textbf{irr}(S ; Fsb? ; mo ; sbF?)
\textbf{irr}(S ; Fsb? ; rf^{-1} ; mo ; sbF?)
SC axioms

\[ \text{irr}(S; Fsb?; hb; sbF?) \]
\[ \text{irr}(S; Fsb?; mo; sbF?) \]
\[ \text{irr}(S; Fsb?; rf^{-1}; mo; sbF?) \]
SC axioms

\[ \text{irr}(S ; Fsb? ; hb ; sbF?) \]

\[ \text{irr}(S ; Fsb? ; \textcolor{blue}{mo} ; sbF?) \]

\[ \text{irr}(S ; Fsb? ; \textcolor{red}{fr} ; sbF?) \]
SC axioms

\textbf{irr}(S ; Fsb? ; hb ; sbF?)

\textbf{irr}(S ; Fsb? ; \textcolor{blue}{mo} ; sbF?)

\textbf{irr}(S ; Fsb? ; fr ; sbF?)
SC axioms

\texttt{irr}(S ; Fsb? ; (hb \cup \texttt{mo} \cup fr) ; sbF?)
All consistency axioms

\[ \text{irr}(\text{hb}) \]
\[ \text{irr}(\text{rf}^{-1}? ; \text{mo} ; \text{rf}? ; \text{hb}) \]
\[ \text{irr}(\text{rf} ; \text{hb}) \]
\[ \text{empty}((\text{rf} ; [\text{nal}]) \setminus \text{vis}) \]
\[ \text{irr}(\text{rf} \cup (\text{mo} ; \text{mo} ; \text{rf}^{-1}) \cup (\text{mo} ; \text{rf})) \]
\[ \text{irr}(S ; \text{hb}) \]
\[ \text{irr}(S ; \text{Fsb}? ; \text{mo} ; \text{sbF}?) \]
\[ \text{irr}(S ; \text{rf}^{-1} ; [\text{SC}] ; \text{mo}) \]
\[ \text{irr}((S \setminus (\text{mo} ; S)) ; \text{rf}^{-1} ; \text{hbl} ; [W]) \]
\[ \text{irr}(S ; \text{Fsb} ; \text{rf}^{-1} ; \text{mo}) \]
\[ \text{irr}(S ; \text{rf}^{-1} ; \text{mo} ; \text{sbF}) \]
\[ \text{irr}(S ; \text{Fsb} ; \text{rf}^{-1} ; \text{mo} ; \text{sbF}) \]
All consistency axioms

\textbf{irr}(hb)

\textbf{irr}(rf^{-1}? ; \textbf{mo} ; rf? ; hb)

\textbf{irr}(rf ; hb)

\textbf{empty}((rf ; [nal]) \setminus \textit{vis})

\textbf{irr}(rf \cup (mo ; mo ; rf^{-1}) \cup (mo ; rf))

\textbf{irr}(S ; Fsb? ; (hb \cup \textbf{mo} \cup \textbf{fr}) ; sbF?)
Changing the standard

6. There shall be a single total order $S$ on all memory_order_seq_cst operations, consistent with the “happens before” order and modification orders for all affected locations, such that each memory_order_seq_cst operation $B$ that loads a value from an atomic object $M$ observes one of the following values:
   - the result of the last modification $A$ of $M$ that precedes $B$ in $S$, if it exists, or
   - if $A$ exists, the result of some modification of $M$ in the visible sequence of side effects with respect to $B$ that is not memory_order_seq_cst and that does not happen before $A$, or
   - if $A$ does not exist, the result of some modification of $M$ in the visible sequence of side effects with respect to $B$ that is not memory_order_seq_cst.

   [...] 

9. For an atomic operation $B$ that reads the value of an atomic object $M$, if there is a memory_order_seq_cst fence $X$ sequenced before $B$, then $B$ observes either the last memory_order_seq_cst modification of $M$ preceding $X$ in the total order $S$ or a later modification of $M$ in its modification order.

10. For atomic operations $A$ and $B$ on an atomic object $M$, where $A$ modifies $M$ and $B$ takes its value, if there is a memory_order_seq_cst fence $X$ such that $A$ is sequenced before $X$ and $B$ follows $X$ in $S$, then $B$ observes either the effects of $A$ or a later modification of $M$ in its modification order.

11. For atomic operations $A$ and $B$ on an atomic object $M$, where $A$ modifies $M$ and $B$ takes its value, if there are memory_order_seq_cst fences $X$ and $Y$ such that $A$ is sequenced before $X$, $Y$ is sequenced before $B$, and $X$ precedes $Y$ in $S$, then $B$ observes either the effects of $A$ or a later modification of $M$ in its modification order.

[276 words; FK reading ease 41.2]

1. A value computation $A$ of an object $M$ reads before a side effect $B$ on $M$ if $A$ and $B$ are different operations and $B$ follows, in the modification order of $M$, the side effect that $A$ observes.

2. If $X$ reads before $Y$, or happens before $Y$, or precedes $Y$ in modification order, then $X$ (as well as any fences sequenced before $X$) is SC-before $Y$ (as well as any fences sequenced after $Y$).

3. If $A$ is SC-before $B$, and $A$ and $B$ are both memory_order_seq_cst, then $A$ is restricted-SC-before $B$.

4. There must be no cycles in restricted-SC-before.

[93 words; FK reading ease 73.1]
Consistent executions

- Execution $X$ is consistent iff there exists $rf$, $mo$ and $S$ such that $(X, rf, mo, S)$ is well-formed and satisfies all the consistency axioms.
SC axioms

\[ \text{irr}(S ; Fsb? ; (hb \cup \text{mo} \cup \text{fr}) ; sbF?) \]

\[ \text{acyclic}(SC^2 \setminus \text{id} \cap (Fsb? ; (hb \cup \text{mo} \cup \text{fr}) ; sbF?)) \]
We also provide performance results gathered using CDSChecker, a custom-built simulator for the C11 memory model [29], which is general-purpose and exhaustive for translating C11 and OpenCL programs into their executions.

Our overhaul of SC atomics avoids the requirement for the relaxed behaviour. We do not need further synchronisation between the loads. However, placing a store(x,1); after the store and an atomic_thread_fence() can lead to improved efficiency in the process of enumerating the multi-copy atomic write(x,1); and OpenCL litmus tests (§6.1), and present experimental results for comparing the performance of both the original SC axioms and (b) when equipped with our revised SC axioms. In different work-groups, so they have different L1 caches), then the logarithmic y-axis, the performance of both is strong enough to rule out this relaxed behaviour.

Figure 1. Simulation time /s

- **total S**: CDSChecker
- **partial S**: CDSChecker

The original memory model, because CDSChecker, unlike other such as DPOR, that make it difficult to assess the soundness and completeness of the tool. It happens that CDSChecker, unlike other such as DPOR, that make it difficult to assess the soundness and completeness of the tool. It happens that CDSChecker, unlike other such as DPOR, that make it difficult to assess the soundness and completeness of the tool.
Outline

• Introduction to the C11 memory model
• Overhauling the rules for SC atomics in C11
• Introduction to the OpenCL memory model
• Overhauling the rules for SC atomics in OpenCL
OpenCL memory regions
OpenCL memory scopes

device
work-group

store(x)  load(x)

local

global  global_fga
OpenCL memory scopes

device

work-group

store(x,WG)  load(x,WG)

local  L1 cache

global  global_fga
OpenCL memory scopes

- **device**
  - **work-group**
    - `store(x,WG)`
    - `load(x,WG)`
  - `local`
- **global**
- **global_fga**

*not inclusive scopes*
OpenCL memory scopes

device

work-group

store(x,DV)

load(x,DV)

local

inclusive scopes

global

global_fga
Outline

• Introduction to the C11 memory model
• Overhauling the rules for SC atomics in C11
• Introduction to the OpenCL memory model
• Overhauling the rules for SC atomics in OpenCL
SC axioms in OpenCL

- Mostly copied from C11
- "There is a total order $S$, providing...
- every SC operation has **ALL** scope and accesses `global_fga` memory...
- **OR** every SC operation has **DV** scope and does not access `global_fga` memory"
Problems

😢 Can't always tell whether a location is `global` or `global_fga`!

😢 The default, which is `memory_scope_device`, is not always enough!

😢 Non-compositional!

😢 Unnecessarily restrictive!

😢 And too weak anyway!
store(x,DV)

load(x,DV)

global
store(x,DV)

load(x,DV)

store(y,DV)

load(y,DV)

store(z1,REL,ALL)

load(z1,ACQ,ALL)
store(x,DV)
load(x,DV)

load(z2,ACQ,ALL)

store(y,DV)
load(y,DV)

store(z2,REL,ALL)
store(x,DV)

load(x,DV)

store(z1,REL,ALL)

load(z2,ACQ,ALL)

store(z2,REL,ALL)

load(z1,ACQ,ALL)

store(y,DV)

load(y,DV)
store(x, DV)
load(x, DV)
store(y, DV)
load(y, DV)
global
global_fga
SC axiom in OpenCL

every SC operation has **ALL** scope and accesses `global_fga` memory

**OR**

every SC operation has **DV** scope and does not access `global_fga` memory

\[
\text{irr}(S ; ((Fsb? ; (hb \cup mo \cup fr) ; sbF?)))
\]
SC axiom in OpenCL

every SC operation has **ALL** scope and accesses **global_fga** memory **OR**
every SC operation has **DV** scope and does not access **global_fga** memory

\[
\text{irr}(S ; ((Fsb? ; (hb \cup mo \cup fr) ; sbF?) \cap incl)
\]
store(x,DV)

load(x,DV)

store(y,DV)

load(y,DV)

global

global_fga

not inclusive scopes
Changing the standard

The following text is reproduced verbatim from the OpenCL 2.1 standard.

### B. Rules for SC atomics in OpenCL

**Local memory** observes one of the following values:

- All `memory_order_seq_cst` operations have the scope `memory_scope_all_svm_devices` and all affected memory locations are contained in system allocations or fine grain SVM buffers with atomics support.
- All `memory_order_seq_cst` operations have the scope `memory_scope_device` and all affected memory locations are not located in system allocated regions or fine-grain SVM buffers with atomics support.

If the total order of an atomic object `M` that reads a value from an atomic object `X` in modification order, then there shall exist a single total order `S` for all `memory_order_seq_cst` operations that is consistent with the modification orders for all affected locations, as well as the appropriate global-happens-before and local-happens-before orders for those locations, such that each `memory_order_seq_cst` operation `B` that reads a value from an atomic object `M` in global or local memory observes one of the following values:

1. The result of the last modification `A` of `M` that precedes `B` in `S`, if it exists, or
2. If `A` exists, the result of some modification of `M` in the visible sequence of side effects with respect to `B` that is not `memory_order_seq_cst` and that does not happen before `A`, or
3. If `A` does not exist, the result of some modification of `M` in the visible sequence of side effects with respect to `B` that is not `memory_order_seq_cst`.

If the total order `S` exists, the following rules hold:

- For an atomic operation `B` that reads the value of an atomic object `M`, if there is a `memory_order_seq_cst` fence `X` sequenced-before `B`, then `B` observes either the last `memory_order_seq_cst` modification of `M` preceding `X` in the total order `S` or a later modification of `M` in its modification order.
- For atomic operations `A` and `B` on an atomic object `M`, where `A` modifies `M` and `B` takes its value, if there is a `memory_order_seq_cst` fence `X` such that `A` is sequenced-before `X` and `B` follows `X` in `S`, then `B` observes either the effects of `A` or a later modification of `M` in its modification order.
- For atomic operations `A` and `B` on an atomic object `M`, where `A` modifies `M` and `B` takes its value, if there are `memory_order_seq_cst` fences `X` and `Y` such that `A` is sequenced-before `X`, `Y` is sequenced-before `B`, and `X` precedes `Y` in `S`, then `B` observes either the effects of `A` or a later modification of `M` in its modification order.
- For atomic operations `A` and `B` on an atomic object `M`, if there are `memory_order_seq_cst` fences `X` and `Y` such that `A` is sequenced-before `X`, `Y` is sequenced-before `B`, and `X` precedes `Y` in `S`, then `B` occurs later than `A` in the modification order of `M`.

If one of the following two conditions holds:

- All `memory_order_seq_cst` operations have the scope `memory_scope_all_svm_devices` and all affected memory locations are contained in system allocations or fine grain SVM buffers with atomics support.
- All `memory_order_seq_cst` operations have the scope `memory_scope_device` and all affected memory locations are not located in system allocated regions or fine-grain SVM buffers with atomics support.

1. A value computation `A` of an object `M` reads before a side effect `B` on `M` if `A` and `B` are different operations and `B` follows, in the modification order of `M`, the side effect that `A` observes.
2. If `X` reads before `Y`, or `Y` reads before `X`, or any fences sequenced before `X` is SC-before `Y` (as well as any fences sequenced after `Y`).
3. If `A` is SC-before `B`, and `A` and `B` have inclusive scopes, then `A` is restricted-SC-before `B`.
4. There must be no cycles in restricted-SC-before.
In short

• The rules for sequentially-consistent atomic operations and fences ("SC atomics") in C11 and OpenCL are 😞 too complex, 😞 too weak, and 😞 too strong.

• We suggest how to fix them 😌.