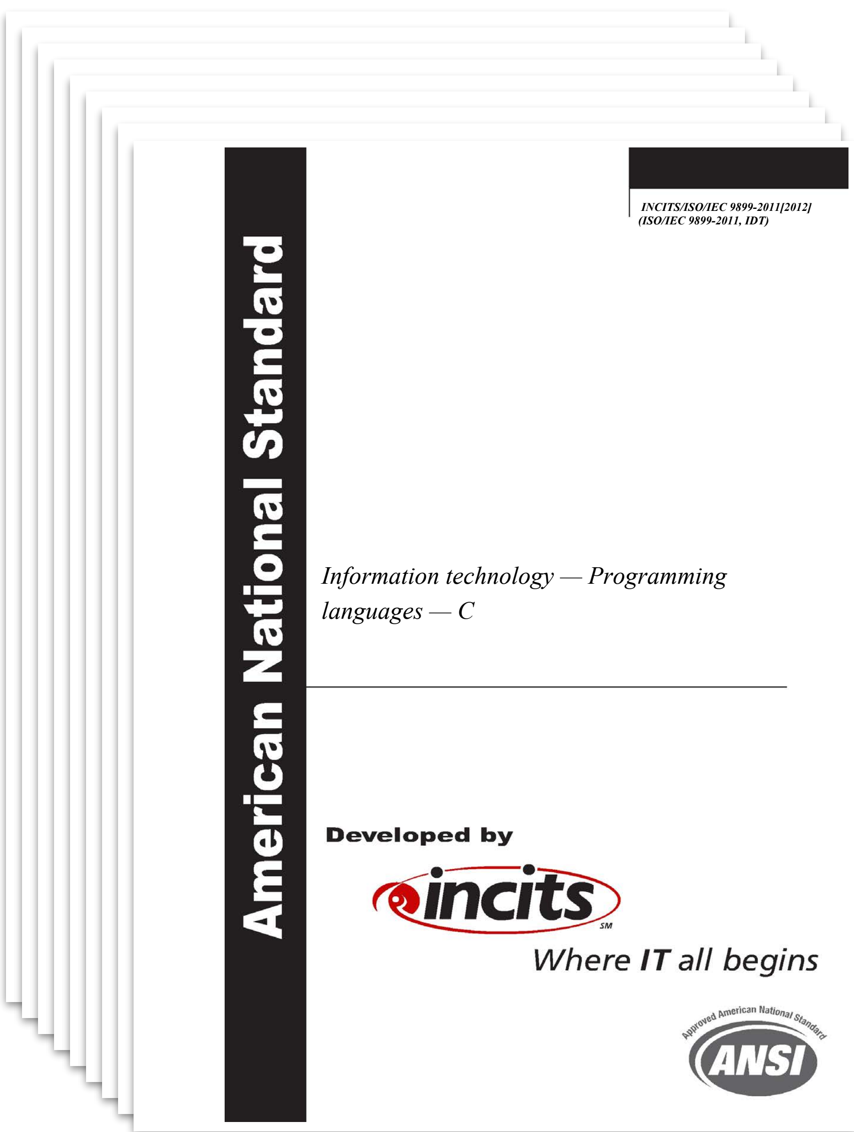
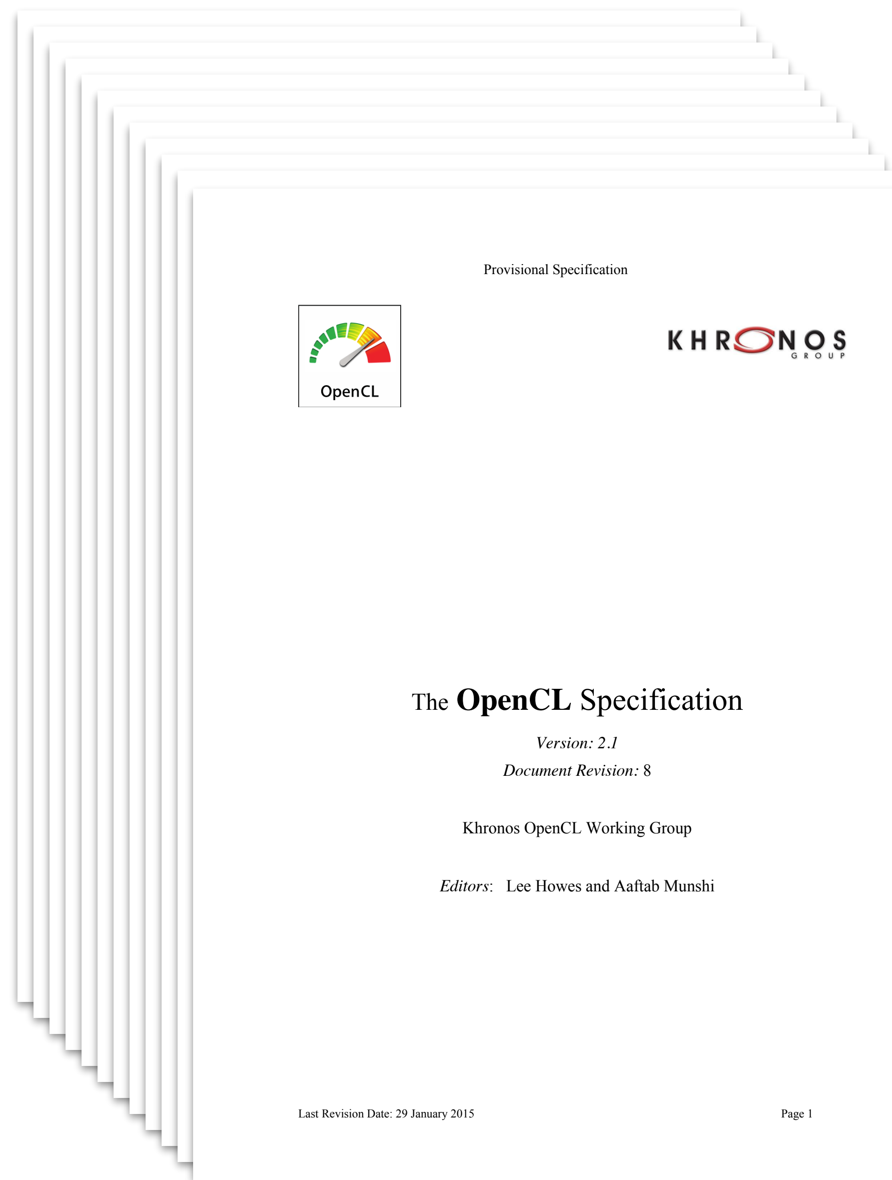
Overhauling SC atomics in C11 and OpenCL

Mark Batty, Alastair F. Donaldson and John Wickerson





S-REPLS @ Middlesex

November 2015

In short

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

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 😌.

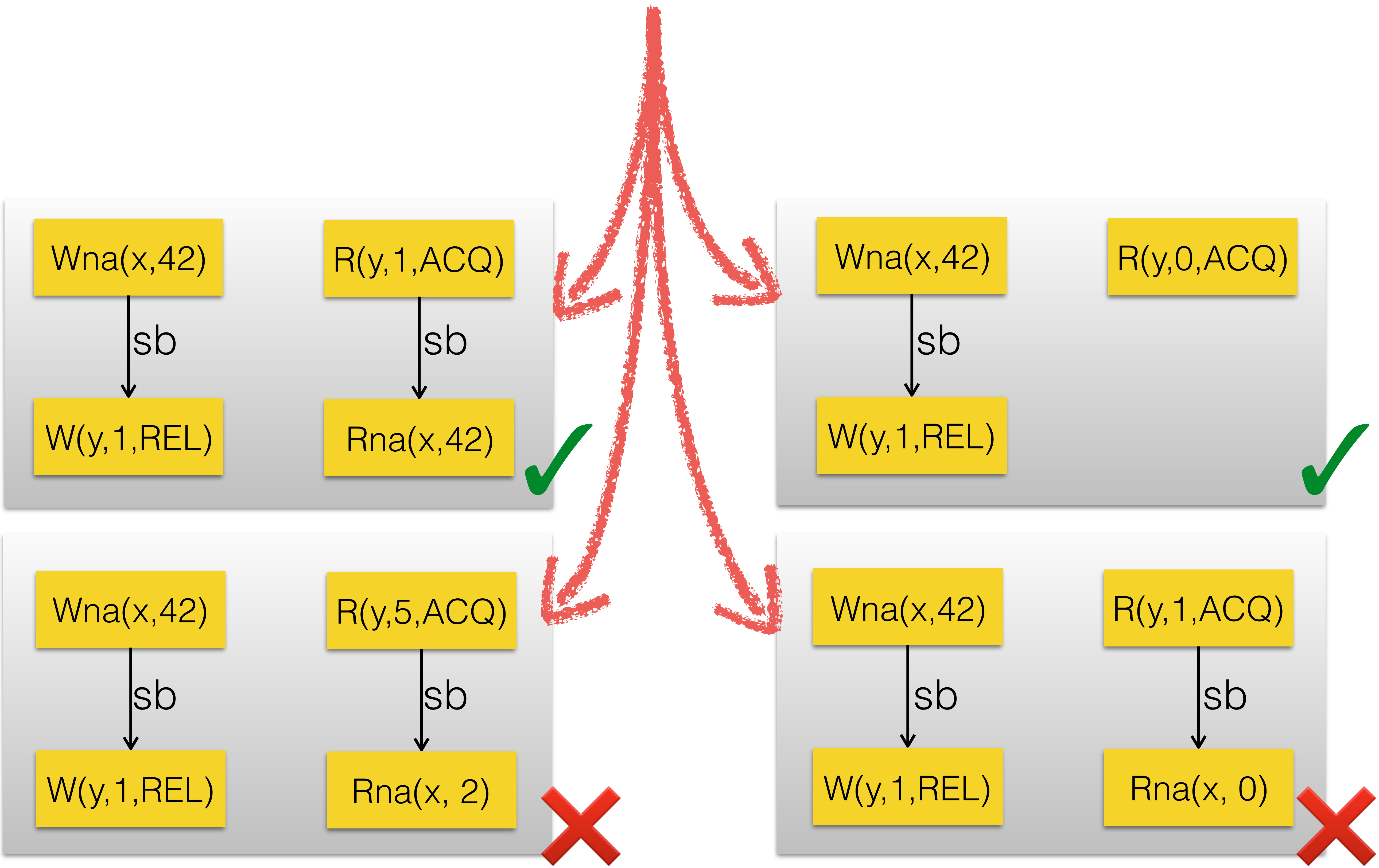
Outline

- 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;                                   ||  if (atomic_load_explicit(y,
atomic_store_explicit(y, 1,                ||        memory_order_acquire))
  memory_order_release);                   ||    print(*x);
```



Consistent executions

- Execution X is **consistent** iff

  satisfies all the **consistency axioms**.

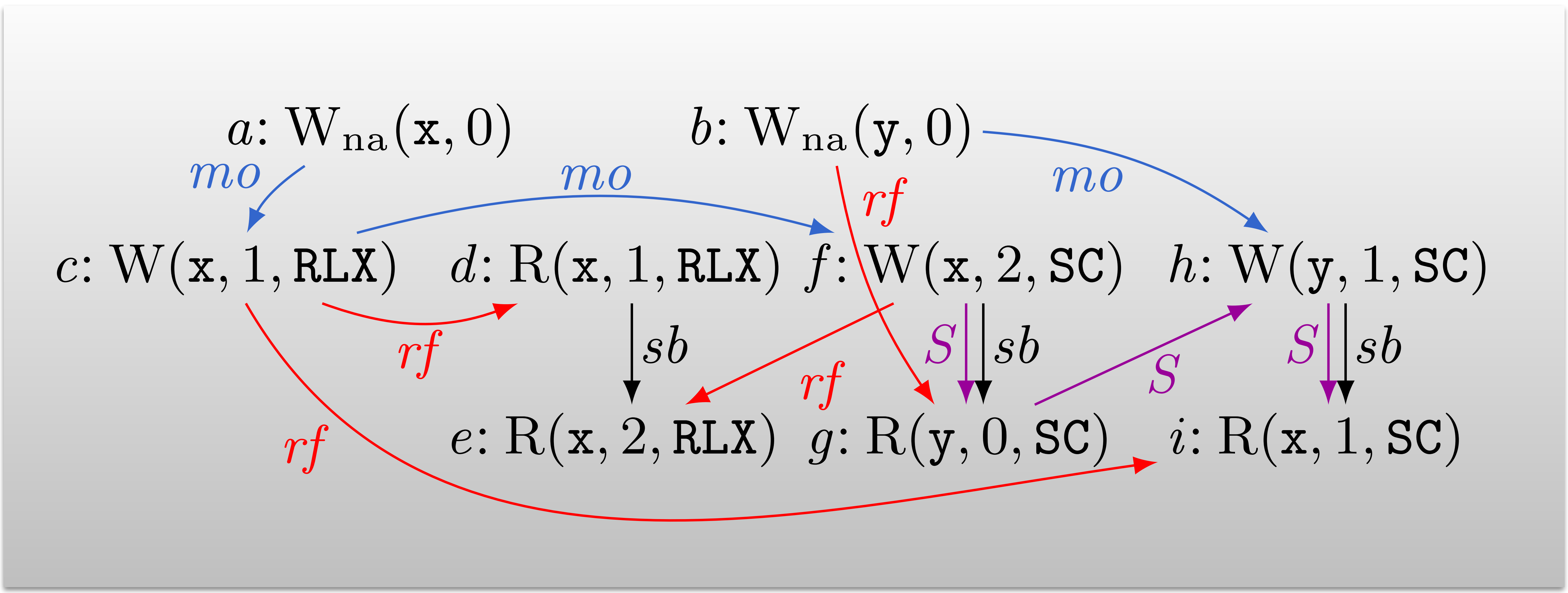
Consistent executions

- 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

$$\begin{tikzcd}
& a\colon \mathrm{W}_{\mathrm{na}}(\mathtt{x}, 0) & b\colon \mathrm{W}_{\mathrm{na}}(\mathtt{y}, 0) & \\
c\colon \mathrm{W}(\mathtt{x}, 1, \mathtt{RLX}) & d\colon \mathrm{R}(\mathtt{x}, 1, \mathtt{RLX}) \arrow[d, "sb"] & f\colon \mathrm{W}(\mathtt{x}, 2, \mathtt{SC}) \arrow[d, "sb"] & h\colon \mathrm{W}(\mathtt{y}, 1, \mathtt{SC}) \arrow[d, "sb"] \\
& e\colon \mathrm{R}(\mathtt{x}, 2, \mathtt{RLX}) & g\colon \mathrm{R}(\mathtt{y}, 0, \mathtt{SC}) & i\colon \mathrm{R}(\mathtt{x}, 1, \mathtt{SC})
\end{tikzcd}$$

Candidate executions



All consistency axioms

irr(hb)

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

irr(rf ; hb)

empty((rf ; [nal]) \ vis)

irr(rf ∪ (mo ; mo ; rf^{-1}) ∪ (mo ; rf))

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

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

Derived relations

$\mathrm{Fsb} = [F] \,;\, \mathrm{sb}$

$\mathrm{sbF} = \mathrm{sb} \,;\, [F]$

$\mathrm{rs}' = \mathrm{thd} \cup (E^2 \,;\, (R \cap W))$

$\mathrm{rs} = \mathrm{mo} \cap \mathrm{rs}' \setminus ((\mathrm{mo} \setminus \mathrm{rs}') \,;\, \mathrm{mo})$

$\mathrm{sw} = ([\mathrm{rel}] \,;\, \mathrm{Fsb}? \,;\, [W \cap A] \,;\, \mathrm{rs}? \,;\, \mathrm{rf} \,;\, [R \cap A] \,;\, \mathrm{sbF}? \,;\, [\mathrm{acq}]) \setminus \mathrm{thd}$

$\mathrm{hb} = (\mathrm{sb} \cup (I \times -I) \cup \mathrm{sw})^+$

$\mathrm{hbl} = \mathrm{hb} \cap \mathrm{loc}$

$\mathrm{vis} = (W \times R) \cap \mathrm{hbl} \setminus (\mathrm{hbl} \,;\, [W] \,;\, \mathrm{hb})$

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

irr(hb)

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

irr(rf ; hb)

empty((rf ; [nal]) \ vis)

irr(rf ∪ (mo ; mo ; rf^{-1}) ∪ (mo ; rf))

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

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

SC axioms

$\mathbf{irr}(S ; hb)$

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

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

$\mathbf{irr}((S \setminus (mo ; S)) ; rf^{-1} ; hbl ; [W])$

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

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

$\mathbf{irr}(S ; Fsb ; rf^{-1} ; mo ; sbF)$

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

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

SC axioms

$\mathbf{irr}(S \;;\; hb)$

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

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

$\mathbf{irr}((S \setminus (mo \;;\; S)) \;;\; rf^{-1} \;;\; hbl \;;\; [W])$

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

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

$\mathbf{irr}(S \;;\; Fsb \;;\; rf^{-1} \;;\; mo \;;\; sbF)$

SC axioms

$\mathbf{irr}(S \,;\, hb)$

$\mathbf{irr}(S \,;\, Fsb? \,;\, mo \,;\, sbF?)$

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

$\mathbf{irr}((S \cancel{\setminus (mo \,;\, S)}) \,;\, rf^{-1} \,;\, hbl \,;\, [W])$

$\mathbf{irr}(S \,;\, Fsb \,;\, rf^{-1} \,;\, mo)$

$\mathbf{irr}(S \,;\, rf^{-1} \,;\, mo \,;\, sbF)$

$\mathbf{irr}(S \,;\, Fsb \,;\, rf^{-1} \,;\, mo \,;\, sbF)$

SC axioms

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{hb})$

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{Fsb?} \;;\; \mathrm{mo} \;;\; \mathrm{sbF?})$

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{rf}^{-1} \;;\; \cancel{[\mathrm{SC}]} \;;\; \mathrm{mo})$

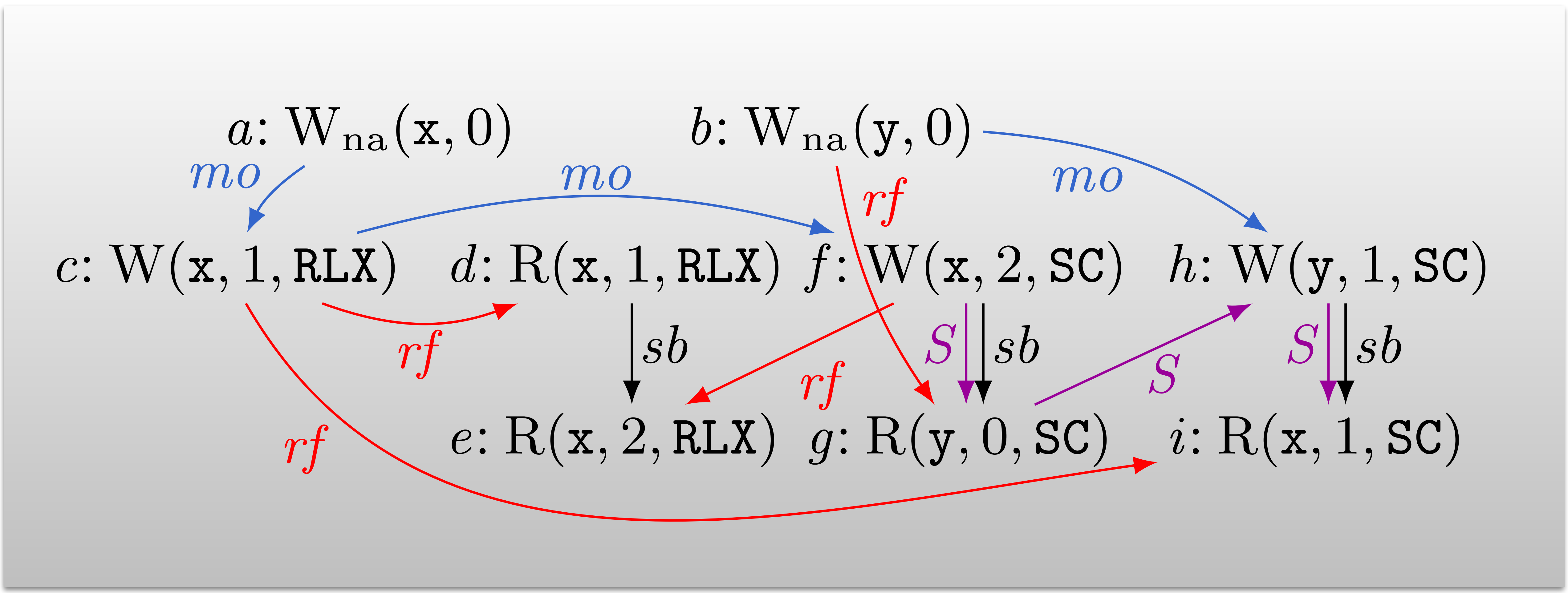
$\mathbf{irr}((\mathrm{S} \cancel{\setminus (\mathrm{mo} \;;\; \mathrm{S})}) \;;\; \mathrm{rf}^{-1} \;;\; \mathrm{hbl} \;;\; [\mathrm{W}])$

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{Fsb} \;;\; \mathrm{rf}^{-1} \;;\; \mathrm{mo})$

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{rf}^{-1} \;;\; \mathrm{mo} \;;\; \mathrm{sbF})$

$\mathbf{irr}(\mathrm{S} \;;\; \mathrm{Fsb} \;;\; \mathrm{rf}^{-1} \;;\; \mathrm{mo} \;;\; \mathrm{sbF})$

Candidate executions



SC axioms

$\mathbf{irr}(S \;;\; hb)$

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

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

$\mathbf{irr}((S \cancel{\setminus (mo \;;\; S)}) \;;\; rf^{-1} \;;\; hbl \;;\; [W])$

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

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

$\mathbf{irr}(S \;;\; Fsb \;;\; rf^{-1} \;;\; mo \;;\; sbF)$

SC axioms

$\mathbf{irr}(S\ ;\ hb)$

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

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

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

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

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

$\mathbf{irr}(S\ ;\ Fsb\ ;\ rf^{-1}\ ;\ mo\ ;\ sbF)$

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

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

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

SC axioms

$\mathbf{irr}(S \;;\; hb)$

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

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

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

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

$\mathbf{irr}(S \;;\; Fsb \;;\; rf^{-1} \;;\; mo \;;\; sbF)$

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

irr(S ; Fsb? ; rf^{-1} ; mo ; sbF?)

SC axioms

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

irr(S ; Fsb? ; rf^{-1} ; mo ; sbF?)

SC axioms

$\mathbf{irr}(S \,;\, hb)$

$\mathbf{irr}(S \,;\, Fsb? \,;\, mo \,;\, sbF?)$

$\mathbf{irr}(S \,;\, Fsb? \,;\, rf^{-1} \,;\, mo \,;\, sbF?)$

SC axioms

irr(S ; Fsb? ; hb ; sbF?)

irr(S ; Fsb? ; mo ; sbF?)

irr(S ; Fsb? ; rf^{-1} ; mo ; sbF?)

SC axioms

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

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

$\mathbf{irr}(S \; ; \; Fsb? \; ; \; rf^{-1} \; ; \; mo \; ; \; sbF?)$

SC axioms

irr(S ; Fsb? ; hb ; sbF?)

irr(S ; Fsb? ; mo ; sbF?)

irr(S ; Fsb? ; fr ; sbF?)

SC axioms

irr(S ; Fsb? ; hb ; sbF?)

irr(S ; Fsb? ; mo ; sbF?)

irr(S ; Fsb? ; fr ; sbF?)

SC axioms

$\mathbf{irr}(S \;;\; Fsb? \;;\; (hb \cup mo \cup fr) \;;\; sbF?)$

All consistency axioms

irr(hb)

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

irr(rf ; hb)

empty((rf ; [nal]) \ vis)

irr(rf ∪ (mo ; mo ; rf^{-1}) ∪ (mo ; rf))

irr(S ; hb)

irr(S ; Fsb? ; mo ; sbF?)

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

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

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

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

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

All consistency axioms

irr(hb)

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

irr(rf ; hb)

empty((rf ; [nal]) \ vis)

irr(rf ∪ (mo ; mo ; rf^{-1}) ∪ (mo ; rf))

irr(S ; Fsb? ; (hb ∪ mo ∪ 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

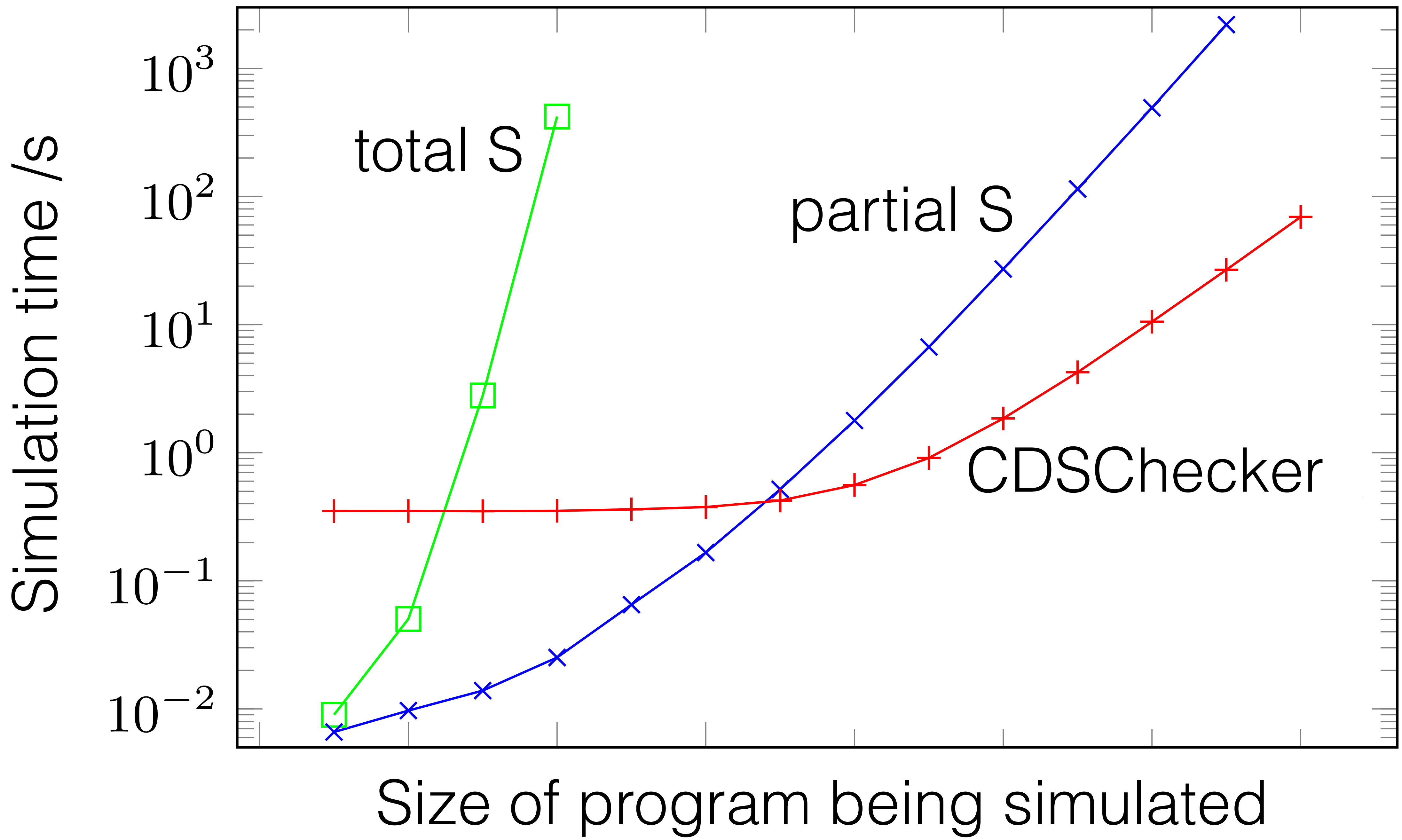
- Execution *X* is **consistent** iff
  there exists *rf*, *mo* and *S* such that
  (*X*,*rf*,*mo*,*S*) is well-formed and
  satisfies all the **consistency axioms**.

SC axioms

$\textbf{irr}(S \,;\, \mathrm{Fsb}? \,;\, (\mathrm{hb} \cup \mathrm{mo} \cup \mathrm{fr}) \,;\, \mathrm{sbF}?)$

$\textbf{acyclic}(\mathrm{SC}^2 \setminus \mathrm{id} \cap (\mathrm{Fsb}? \,;\, (\mathrm{hb} \cup \mathrm{mo} \cup \mathrm{fr}) \,;\, \mathrm{sbF}?))$

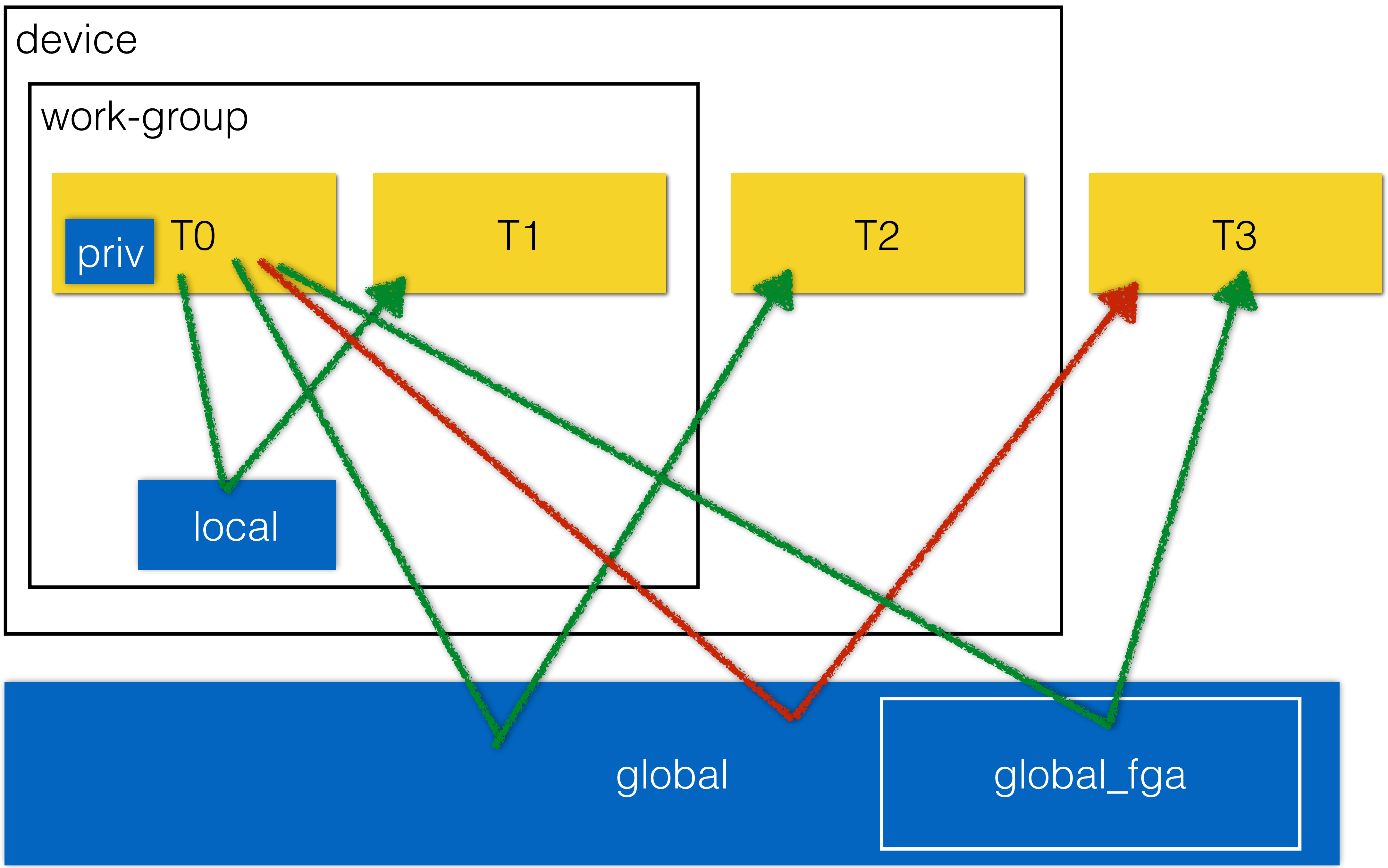
Performance impact



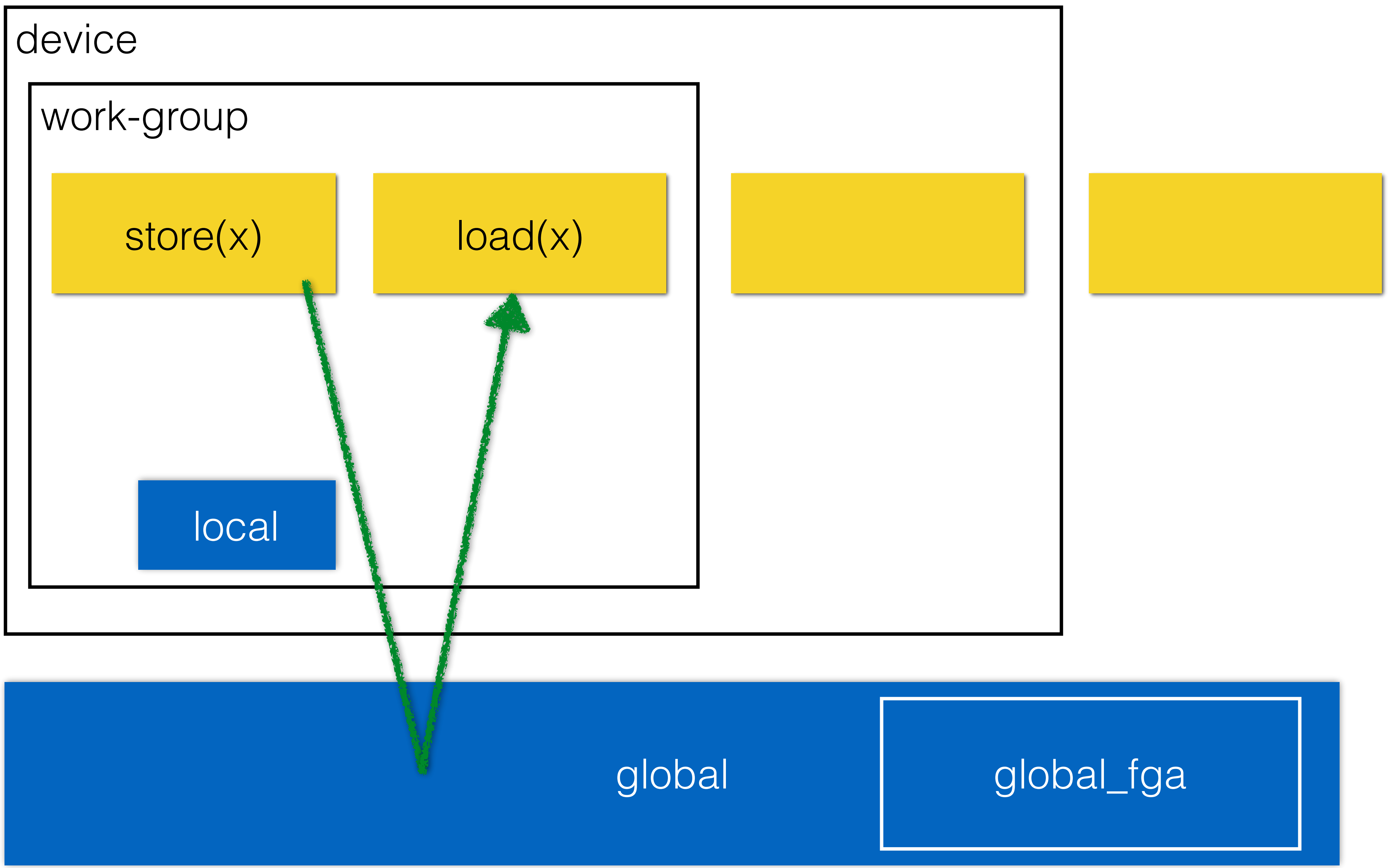
Outline

- 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

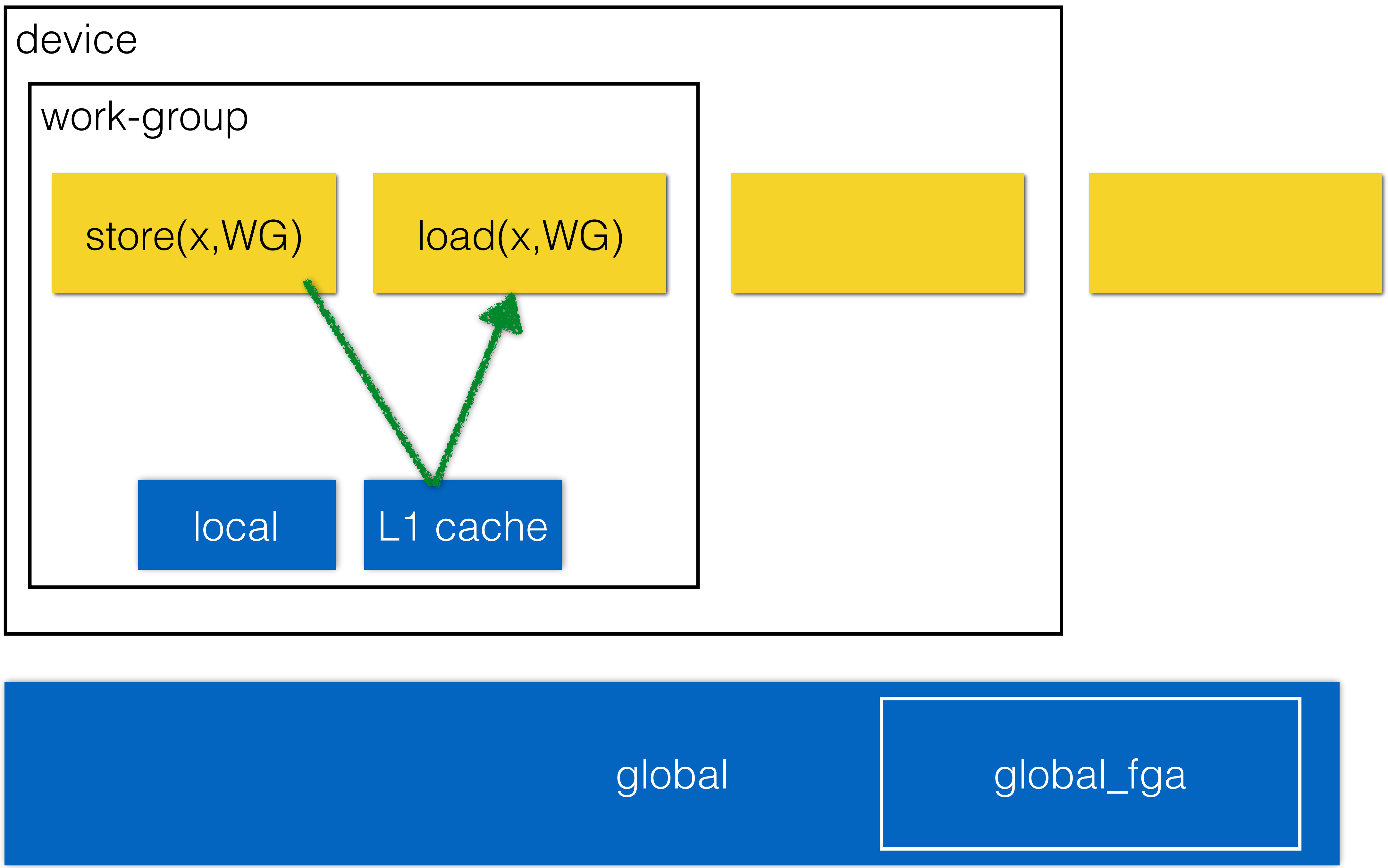
OpenCL memory regions



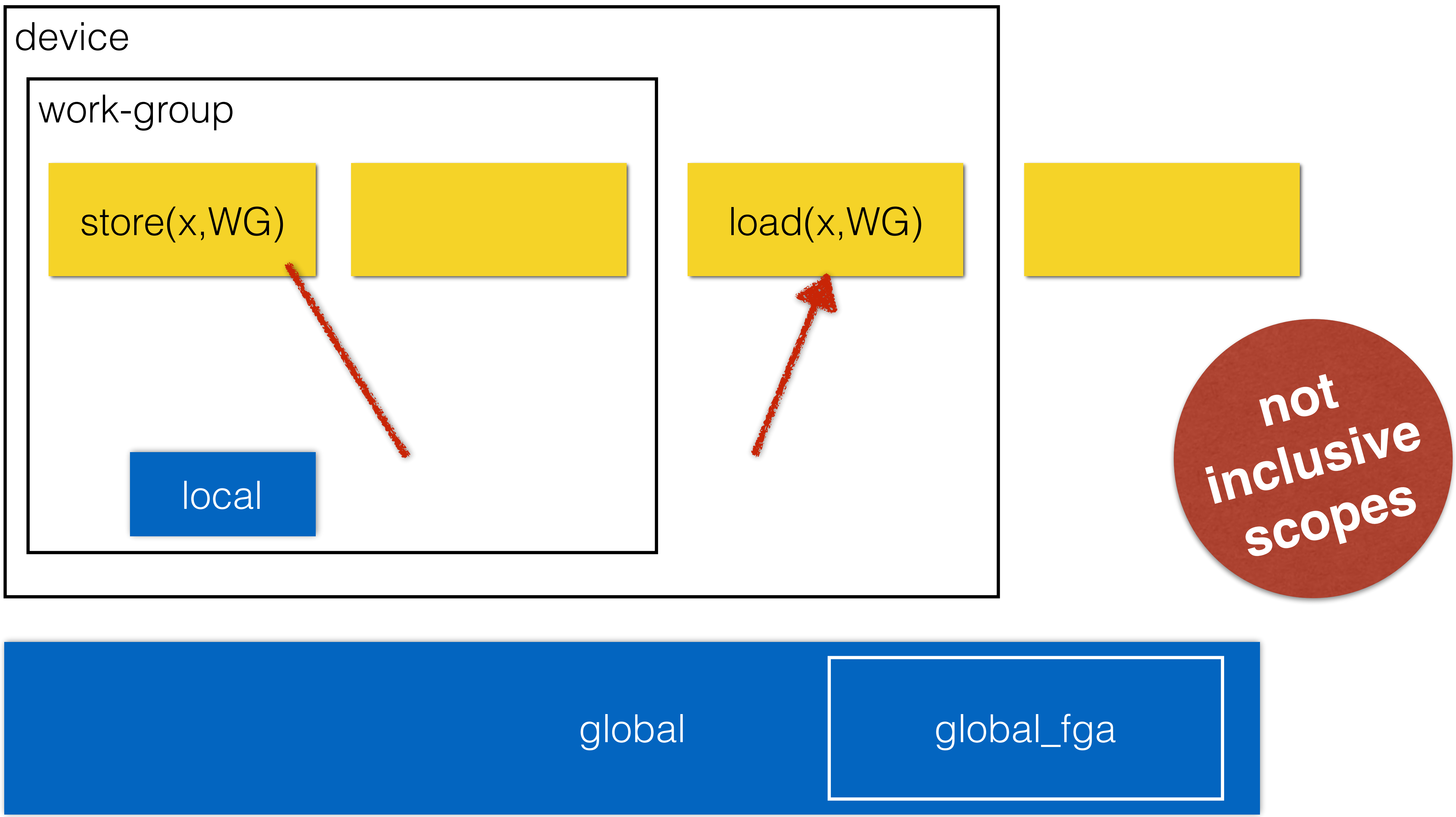
OpenCL memory scopes



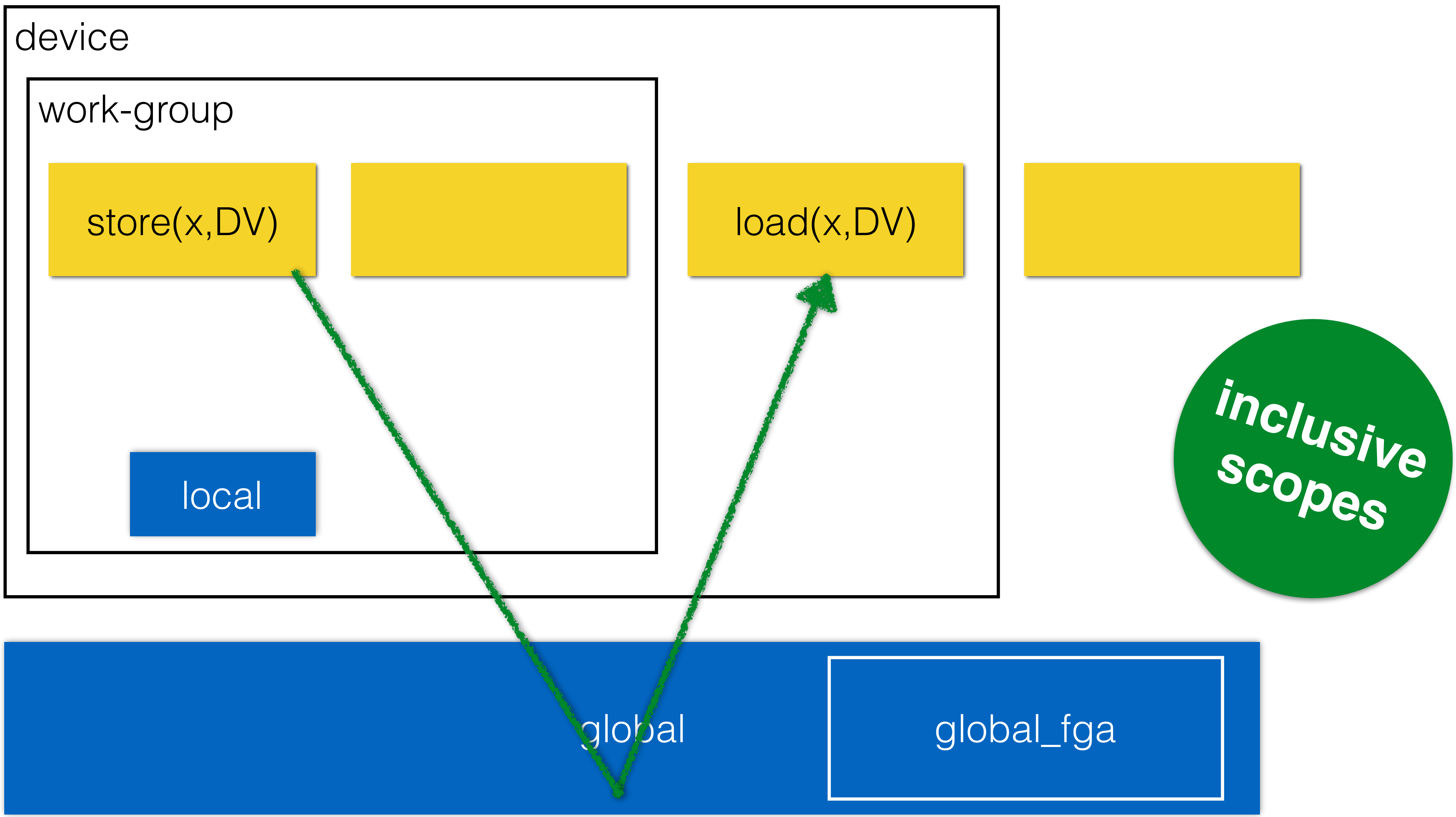
OpenCL memory scopes



OpenCL memory scopes



OpenCL memory scopes

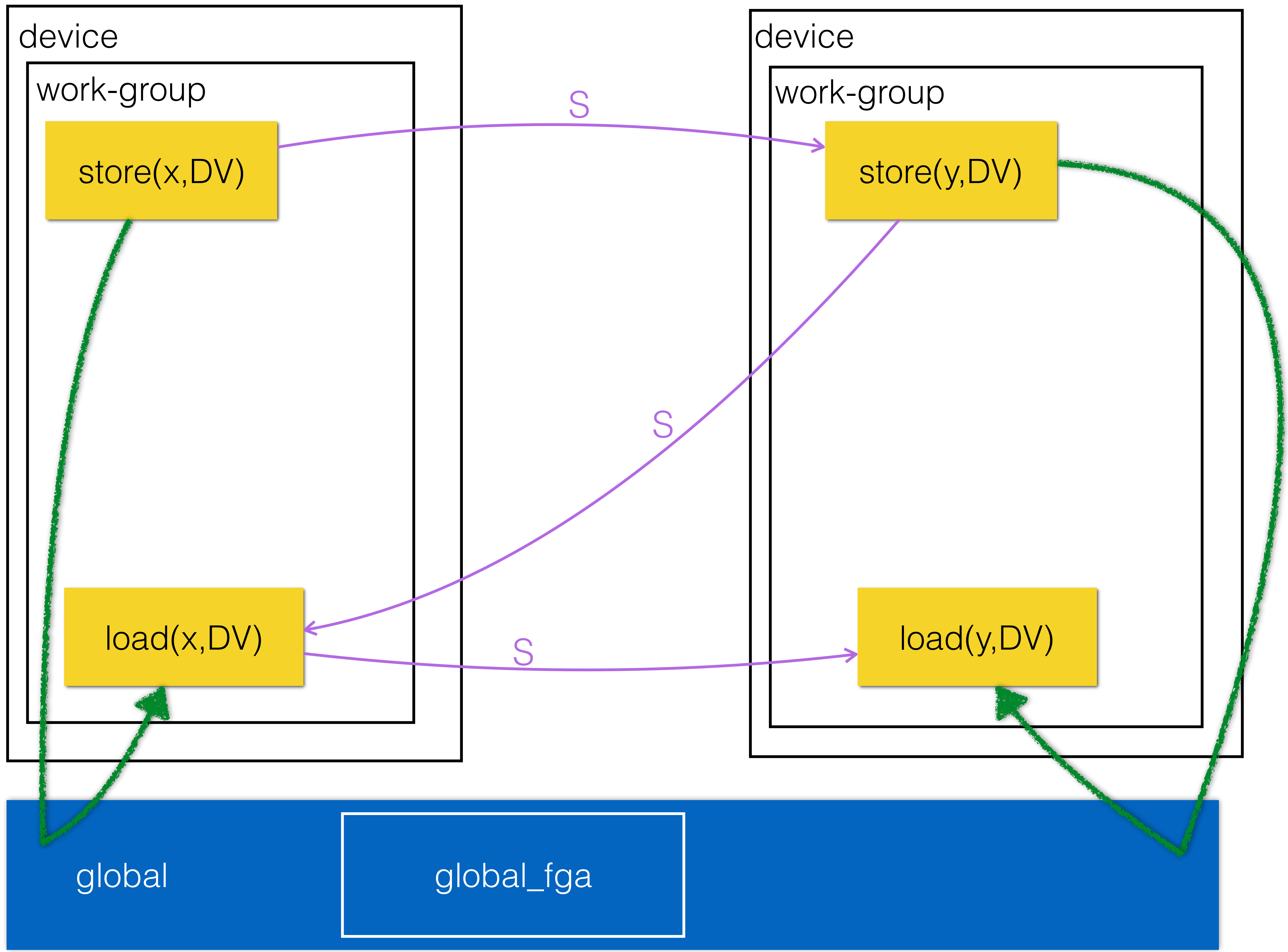


Outline

- 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

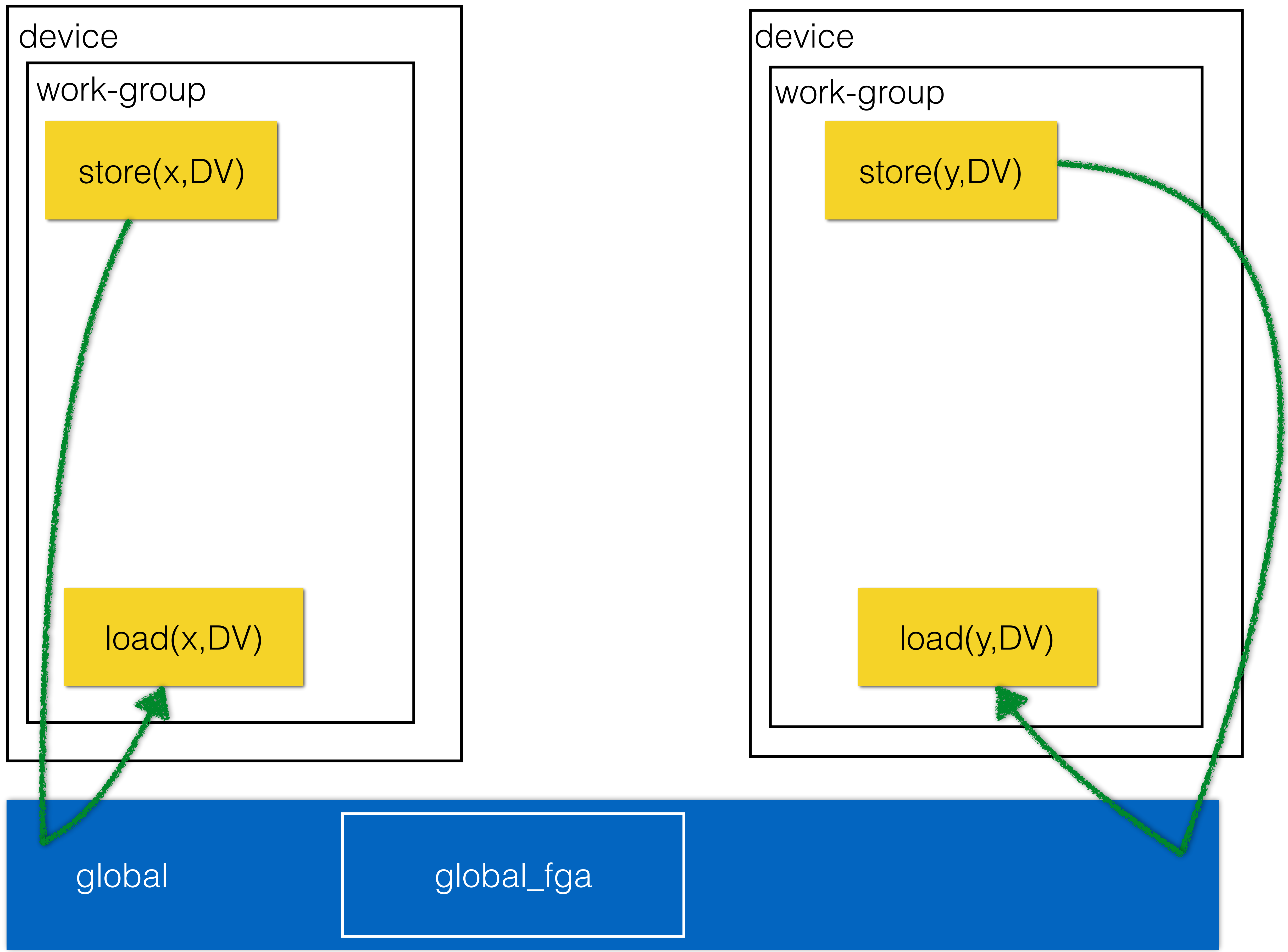
SC axioms in OpenCL

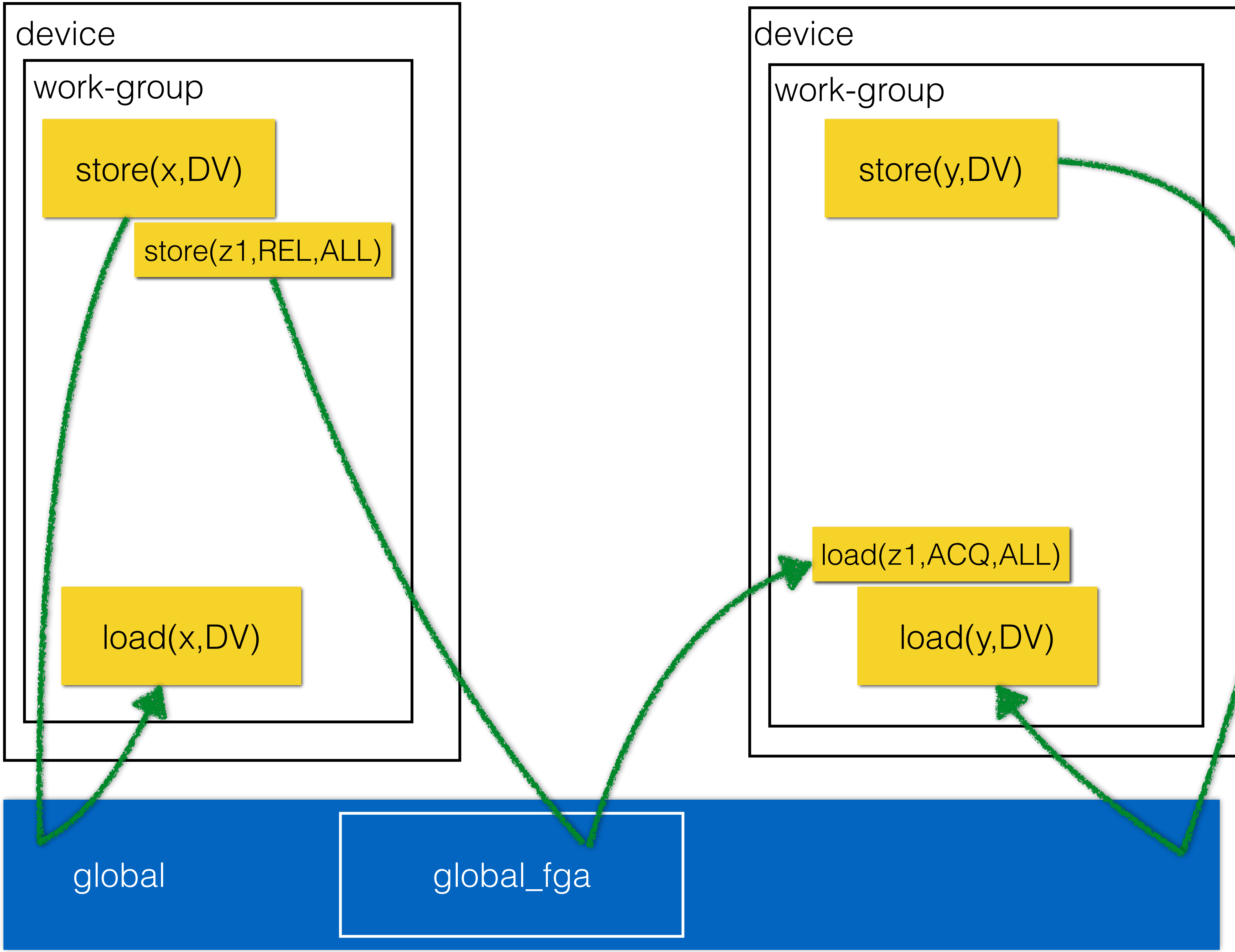
- Mostly copied from C11
- "There is a total order S, providing...
- every SC operation has **`ALL`** scope
  and accesses **`global_fga`** memory...
- **OR** every SC operation has **`DV`** scope
  and does not access **`global_fga`** memory"

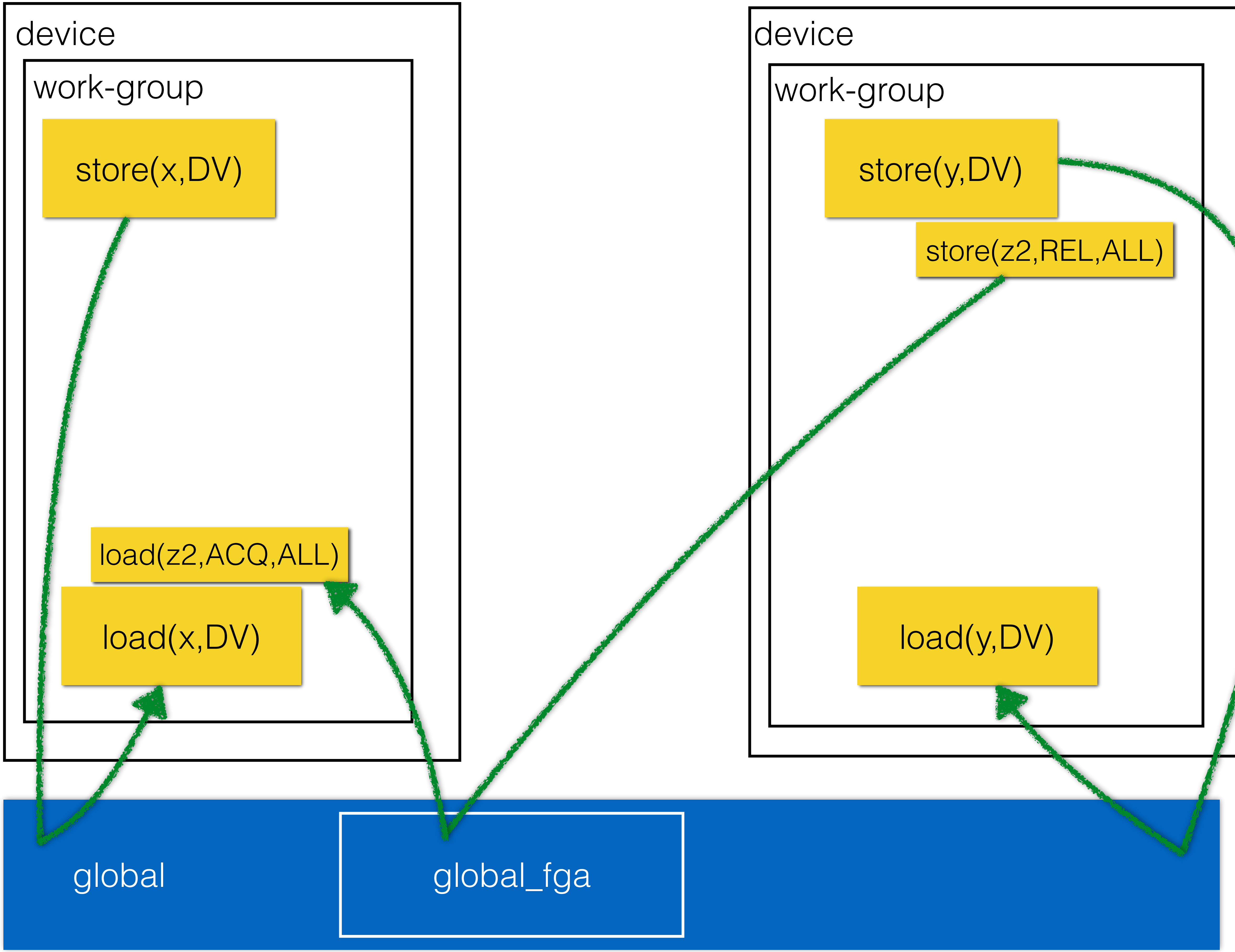

device
work-group
store(x,DV)
load(x,DV)
S
S
S
device
work-group
store(y,DV)
load(y,DV)
global
global_fga

Problems

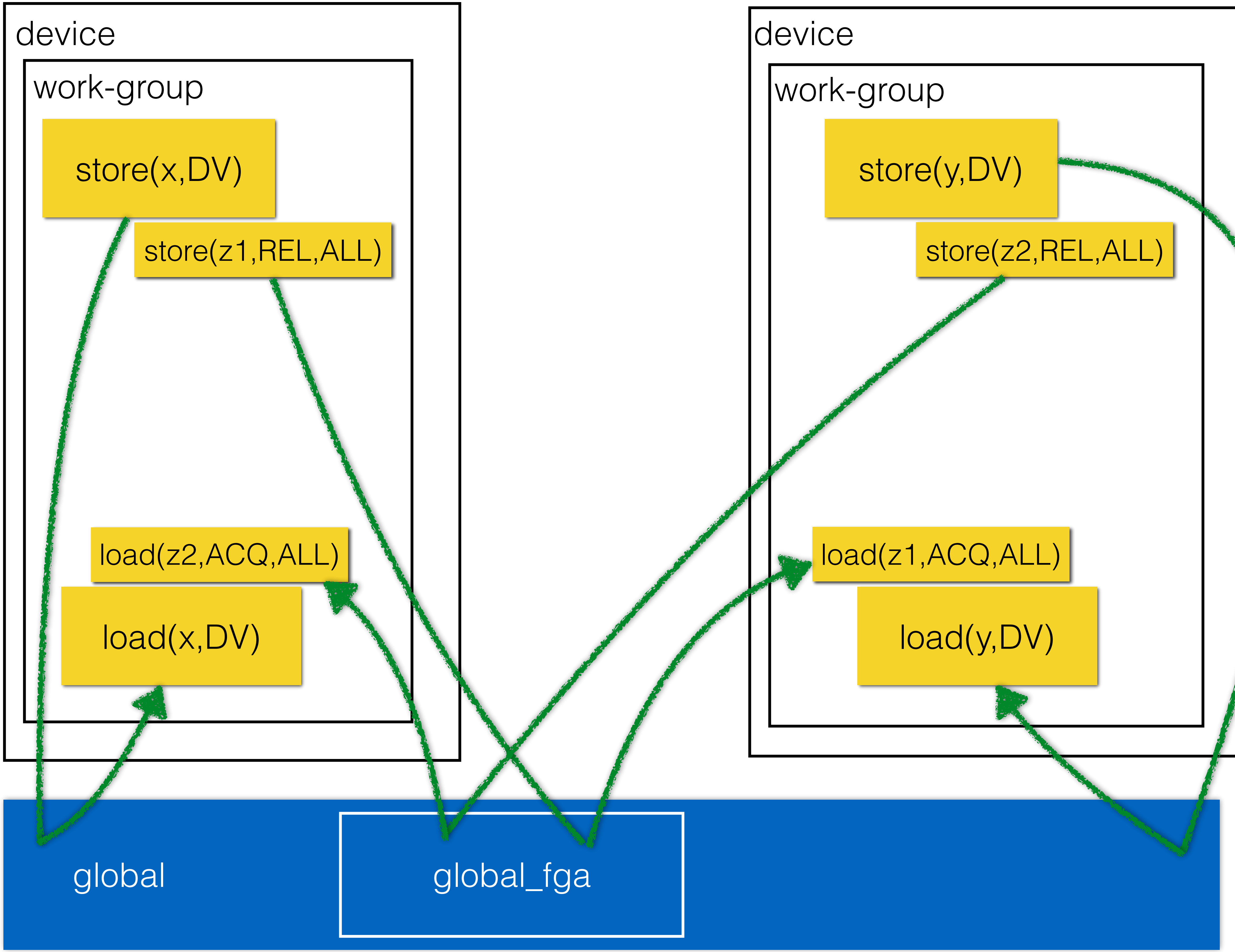
- Can't always tell whether a location is `global` or `global_fga`!
- The default, which is `memory_scope_device`, is not always enough!
- Non-compositional!
- Unnecessarily restrictive!
- And too weak anyway!


device
work-group
store(x,DV)
load(x,DV)
device
work-group
store(y,DV)
load(y,DV)
global
global_fga

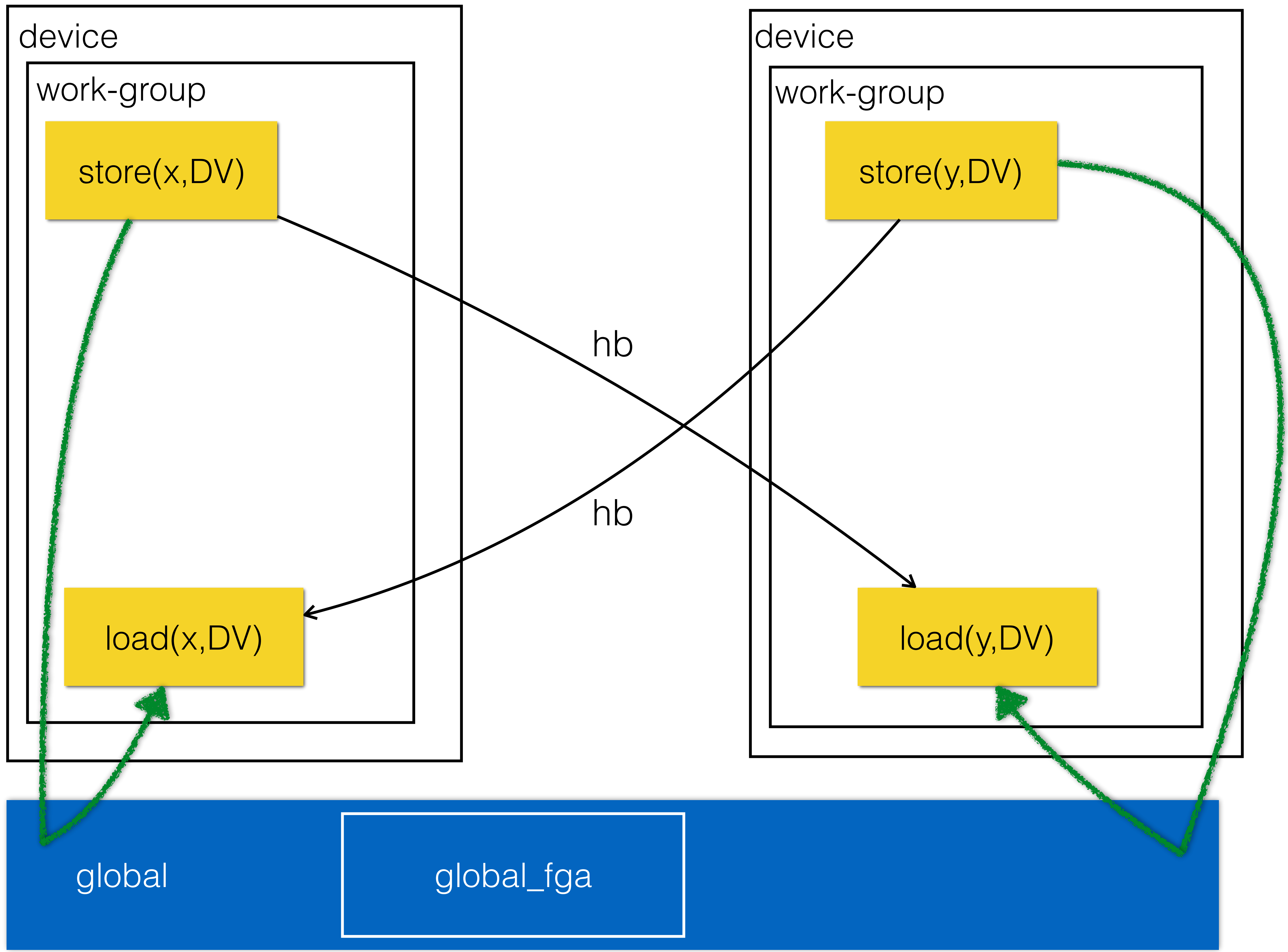

device
work-group
store(x,DV)
store(z1,REL,ALL)
load(x,DV)
device
work-group
store(y,DV)
load(z1,ACQ,ALL)
load(y,DV)
global
global_fga



device
work-group
store(x,DV)
load(z2,ACQ,ALL)
load(x,DV)
device
work-group
store(y,DV)
store(z2,REL,ALL)
load(y,DV)
global
global_fga



device
work-group
store(x,DV)
store(z1,REL,ALL)
load(z2,ACQ,ALL)
load(x,DV)
device
work-group
store(y,DV)
store(z2,REL,ALL)
load(z1,ACQ,ALL)
load(y,DV)
global
global_fga


device
work-group
store(x,DV)
load(x,DV)
device
work-group
store(y,DV)
load(y,DV)
hb
hb
global
global_fga

SC axiom in OpenCL

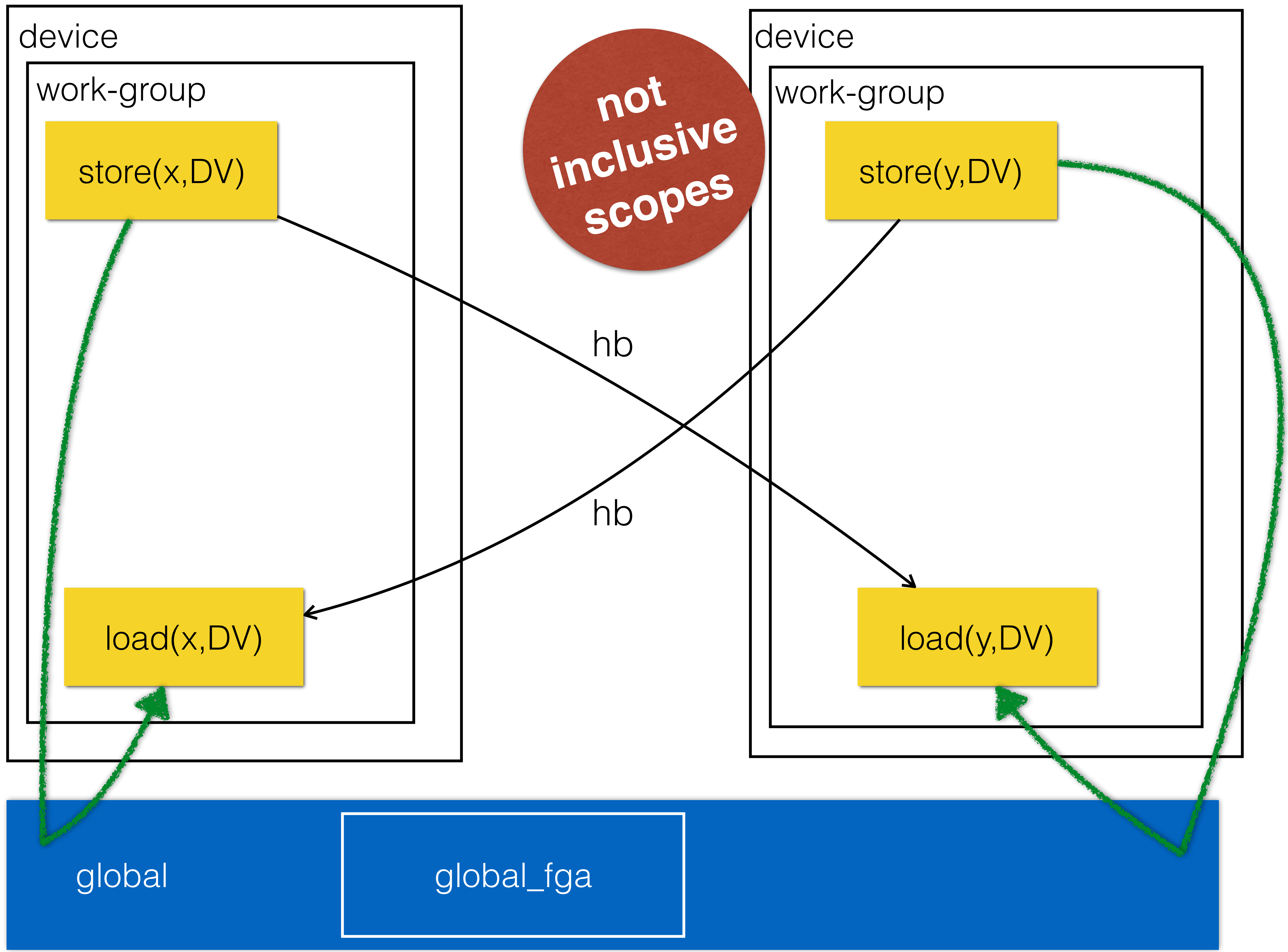
every SC operation has `ALL` scope
and accesses `global_fga` memory
OR
every SC operation has `DV` scope
and does not access `global_fga` memory

irr(S ; ((Fsb? ; (hb ∪ mo ∪ 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~~

irr(S ; ((Fsb? ; (hb ∪ mo ∪ fr) ; sbF?) ∩ incl)


device
work-group
store(x,DV)
load(x,DV)
not inclusive scopes
hb
hb
device
work-group
store(y,DV)
load(y,DV)
global
global_fga

Changing the standard

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

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 loads a value from an atomic object M in global or local memory 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`.

[...]
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.

[391 words; FK reading ease -22.0]

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 global happens before Y, or local 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`, and A and B have inclusive scopes, then A is *restricted-SC-before* B.
4. There must be no cycles in restricted-SC-before.

[106 words; FK reading ease 71.0]

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 😌.