1 MODEL

$$\begin{array}{llll} \mu & ::= \mathsf{wk} & \text{(Weak)} \\ & \mid \mathsf{rlx} & \text{(Relaxed)} \\ & \mid \mathsf{ra} & \text{(Release/Acquire)} \\ & \mid \mathsf{sc} & \text{(Sequentially Consistent)} \end{array} \qquad \begin{array}{lll} \varsigma & ::= \mathsf{cta} & \text{(Thread group)} \\ & \mid \mathsf{gpu} & \text{(Processor)} \\ & \mid \mathsf{sys} & \text{(System)} \end{array}$$

Orders/Relations in model

- $\trianglelefteq$ is the old $\leq$ (without coherence stuff from F4 and P5B).
  This provides the NO-TAR axiom.
- $\leq$ is a the *happens-before* suborder, which only includes rf when they are morally strong.
  This serves as a cross-location transitive kernel for the per-location order.
- $\sqsubseteq$ is a per-location order that relates morally strong and poloc accesses
  This includes $\leq$ for morally strong accesses.
  This provides the SC-PER-LOC axiom.

Write $d \vartriangle e$ if they conflict (ie, read/write or write/write, same location).
Write $d \blacktriangle e$ if they conflict and are morally strong

Definition 1.1. A *pomset with preconditions* is a tuple $(E, \lambda, \leq, \trianglelefteq, \sqsubseteq)$ where

(M1) E is a set of *events*
(M2) $\lambda : E \longrightarrow (\Phi \times \mathcal{A})$ is a *labeling* from which we derive functions
- $\Phi : E \longrightarrow \Phi$ *(formulae)*
- $\mathcal{A} : E \longrightarrow \mathcal{A}$ *(actions)*

(M3) $\leq \subseteq (E \times E)$, $\trianglelefteq \subseteq (E \times E)$, and $\sqsubseteq \subseteq (E \times E)$ are partial orders
(M4) $\bigwedge_e \Phi(e)$ is satisfiable *(consistency)*
(M5) if $d \trianglelefteq e$ then $\Phi(e)$ implies $\Phi(d)$ *(causal strengthening)*
(M6) if $d \leq e$ then $d \trianglelefteq e$
(M7) if $d \leq e$ and d conflicts with e then $d \sqsubseteq e$

We say $d < e$ when $d \leq e$ and $d \neq e$, and similarly for $\vartriangleleft$ and $\sqsubset$.

Definition 1.2 (Strong fulfillment). We say $\mathcal{A}(d) = (\mathsf{W}xv)$ *fulfills* $\mathcal{A}(e) = (\mathsf{R}xv)$ if

(F3A) $d \vartriangleleft e$
(F3B) $d < e$ if d is morally strong with e
(F3C) $d \sqsubseteq e$ (if d is not morally strong with e)
(F4) $\forall \mathcal{A}(c) = (\mathsf{W}x..)$ either $c \sqsubseteq d$ or $e \sqsubseteq c$,

Definition 1.3 (Weak fulfillment). We say $\mathcal{A}(d) = (\mathsf{W}xv)$ *fulfills* $\mathcal{A}(e) = (\mathsf{R}xv)$ if

(F3A) $d \vartriangleleft e$
(F3B) $d < e$ if d is morally strong with e
(F3C) $e \not\sqsubseteq d$ (if d is not morally strong with e)
(F4) $\forall \mathcal{A}(c) = (\mathsf{W}x..)$ either $c \precsim d$ or $e \precsim c$, where

$$d \precsim e \text{ when } \begin{cases} d \sqsubseteq e & \text{if } d \text{ is morally strong with } e \\ e \not\sqsubseteq d & \text{otherwise} \end{cases}$$

If all accesses are morally strong with each other, weak fulfillment degenerates to

(F3) $d < e$
(F4) $\forall \mathcal{A}(c) = (\mathsf{W}x..)$ either $c \sqsubseteq d$ or $e \sqsubseteq c$

If no accesses are morally strong with each other, weak fulfillment degenerates to

(F3) $e \not\sqsubseteq d$
(F4) $\nexists \mathcal{A}(c) = (\mathsf{W}x..)$ both $d \sqsubset c$ and $c \sqsubset e$

Note that the difference between strong and weak fulfillment is limited to $\sqsubseteq$. We sometimes write $\sqsubseteq_{\mathsf{s}}$ for strong fulfillment and $\sqsubseteq_{\mathsf{w}}$ for weak fulfillment.

Prefixing is as in OOPSLA, using $\leq$ for order everywhere except P5B, which has $\sqsubseteq$.

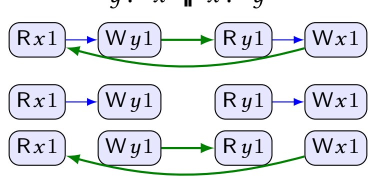
Definition 1.4. Let $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$ when $(\exists P \in \mathcal{P})\ (\forall e \in E)$

(P1) $E' = E \cup \{d\}$
(P2) $\leq' \supseteq \leq$, $\trianglelefteq' \supseteq \trianglelefteq$, and $\sqsubseteq' \supseteq \sqsubseteq$
(P3A) $\mathcal{A}'(e) = \mathcal{A}(e)$
(P3B) $\mathcal{A}'(d) = a$
(P4A) $\Phi'(d)$ implies $\phi \wedge (d \notin E \vee \Phi(d))$
(P4B) if $d \neq (\mathsf{R}..)$ then $e = d$ or $\Phi'(e)$ implies $\Phi(e)$
(P4C) if $d = (\mathsf{R}vx)$ then $e = d$ or $\Phi'(e)$ implies $\Phi(e)[v/x]$
(P5A) if $d = (\mathsf{R}..)$, $e = (\mathsf{W}..)$ then $e = d$ or $\Phi'(e)$ implies $\Phi(e)$ or $d \leq' e$
(P5B) if d conflicts with e then $d \sqsubseteq' e$
(P5C) if d is an acquire or e is a release then $d \leq' e$
(P5D) if d is an SC write and e is an SC read then $d \leq' e$
(P5E) if d reads, and e is an acquiring fence, then $d \leq' e$
(P5F) if d is a releasing fence, and e writes, then $d \leq' e$

2 THIN AIR

Need $\trianglelefteq$ to prevent thin air on rlx:

$$y := x \parallel x := y$$



($\trianglelefteq$)

($\leq$)

($\sqsubseteq_{\mathsf{s}}$)

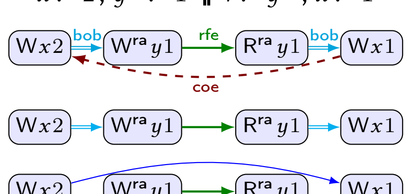
3 IMM EXAMPLES

Interpreting this definition for the IMM:

- No wk, default is rlx
- All threads in same cta (only one scope)
- Actions are morally strong when both are ra/sc, mimicking happens-before
- Strong fulfillment may do the right thing

Disallowed by IMM:

$$x := 2;\ y^{\mathsf{ra}} := 1 \parallel r := y^{\mathsf{ra}};\ x := 1$$ (PUB-REL-ACQ-COE)



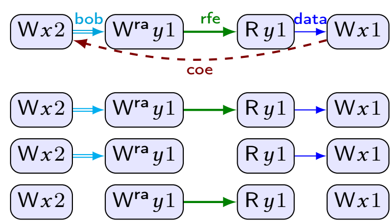
(✗IMM)

($\trianglelefteq = \leq$)

($\sqsubseteq_{\mathsf{s}}$)

Allowed by IMM, but not by Power/ARMv7/ARMv8/TSO:

$$x := 2;\ y^{\mathsf{ra}} := 1 \parallel r := y;\ x := 1 \qquad (\textsc{pub-rel-rlx-coe})$$



(✓IMM)

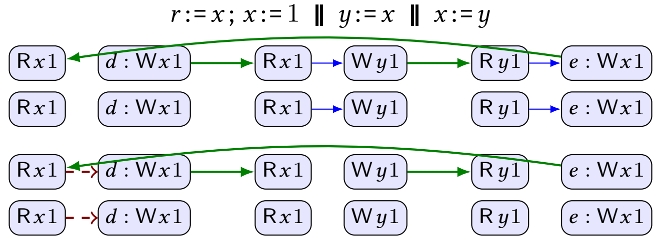
(⊴)

(≤)

(⊑)

Example from talk:

$$r := x;\ x := 1 \parallel y := x \parallel x := y \qquad (\textsc{arm7-weak})$$



(⊴)

(≤)

(⊑)

(⊑ with W)

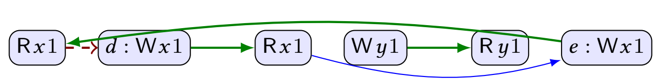
4 TWO ORDER IDEA

The two order idea from OOPSLA talk is:

- Require: $d \sqsubseteq e$ when $d \trianglelefteq e$ and they conflict

This does not work for the IMM or ARMv7, but it may work for Power, TSO, ARMv8. That would be nice. Let's write ⊑ for this notion, with strong fulfillment.

With this there is a cycle in ARM7-WEAK (weak/strong fulfillment not relevant here):



(⊑)

Anton says: ARM7-WEAK is forbidden by Power, TSO, ARMv8, but allowed by ARMv7. Maybe it isn't that important to support it anymore.

There is also a cycle in PUB-REL-RLX-COE. Anton says: I checked Power/ARMv7 models in this regard. They disallow the behavior (as well as ARMv8 and TSO), so we can in principle strengthen IMM to forbid it as well. For that, we may add axiom to IMM forbidding cycles in $\mathsf{co} \cup ([\mathsf{W}]; \mathsf{rfe}^?; ([\mathsf{R}^{\mathsf{acq}}] \cup \mathsf{po}; [\mathsf{FW}^{\mathsf{rel}}]); \mathsf{ar}^*; [\mathsf{W}])$. This works if we have acquire/release accesses on the path since they are compiled with fences to Power.

5 PTX EXAMPLES

Based on [Lustig et al. 2019; NVIDIA 2020].

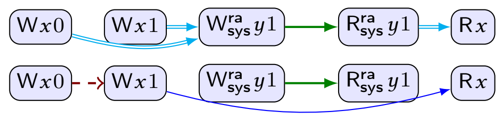
PTX requires weak fulfillment.

Default scope is cta. In examples, all threads in different ctas.

Default mode is wk.

(Rx0) must be forbidden. Before fulfilling the read:

$$x := 0;\ x := 1;\ y^{\mathsf{ra}}_{\mathsf{sys}} := 1 \ \| \ r := y^{\mathsf{ra}}_{\mathsf{sys}};\ s := x \qquad (\text{PUB1}_{\text{SYS}})$$



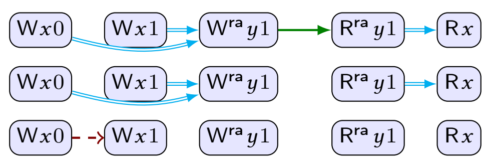
($\trianglelefteq$ = $\leq$)

($\sqsubseteq$)

(Wx1) $\sqsubsetneq$ (Rx) is required by M7, enforcing publication.

(Rx0) must be allowed:

$$x := 0;\ x := 1;\ y^{\mathsf{ra}} := 1 \ \| \ r := y^{\mathsf{ra}};\ s := x \qquad (\text{PUB1}_{\text{CTA}})$$



($\trianglelefteq$)

($\leq$)

($\sqsubseteq$)

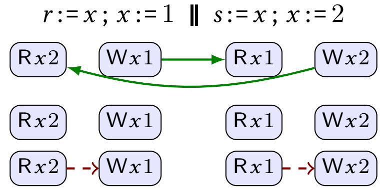
We do not have (W$^{\mathsf{ra}}y$1) $\leq$ (R$^{\mathsf{ra}}y$1) since F3 only requires order for things that are morally strong.

Another example that may be of interest (nothing morally strong). Can this (Rx0)?

$$x := 0;\ x := 1 \ \| \ y := x \ \| \ \mathtt{if}(y)\{r := x\}$$

PTX allows TC16 for events that are not mutually strong (TC16$_{\text{WK}}$), but disallows it when events are mutually strong (TC16$_{\text{SYS}}$). Note that $\leq$ imposes no requirements here. Fulfillment imposes no order. This example shows that F3C cannot be strengthened to require that $d \sqsubseteq e$.

$$r := x;\ x := 1 \ \| \ s := x;\ x := 2 \qquad (\text{TC16}_{\text{WK}})$$

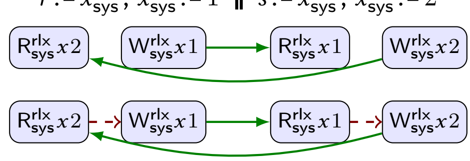


($\trianglelefteq$)

($\leq$)

($\sqsubseteq$)

$$r := x^{\mathsf{rlx}}_{\mathsf{sys}};\ x^{\mathsf{rlx}}_{\mathsf{sys}} := 1 \ \| \ s := x^{\mathsf{rlx}}_{\mathsf{sys}};\ x^{\mathsf{rlx}}_{\mathsf{sys}} := 2 \qquad (\text{TC16}_{\text{SYS}})$$

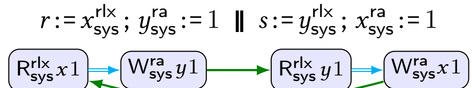


($\trianglelefteq$ = $\leq$)

($\sqsubseteq$)

About Release-Acquire semantics. Anton confirms that the following example is allowed in C11, but disallowed in the IMM. It is apparently allowed in C11 with the intention to allow releasing writes to be downgraded to relaxed in the case that only fulfill relaxed reads.

$$r := x^{\mathsf{rlx}}_{\mathsf{sys}};\ y^{\mathsf{ra}}_{\mathsf{sys}} := 1 \ \| \ s := y^{\mathsf{rlx}}_{\mathsf{sys}};\ x^{\mathsf{ra}}_{\mathsf{sys}} := 1 \qquad (\text{LB-REL})$$

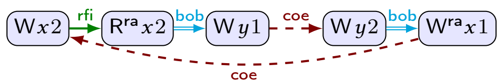


($\trianglelefteq$ = $\leq$)

6 RFI EXAMPLES

Anton example 1 (Allowed by ARM) `[rfi-coe-coe]`

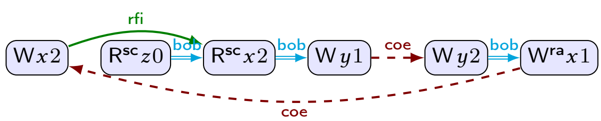
$$x := 2;\ r := x^{\mathsf{ra}};\ y := 1 \parallel y := 2;\ x^{\mathsf{ra}} := 1 \qquad \text{(RFI-COE-COE)}$$



(✓ARM8)

Internal reads survive acquires `[rfi-acq-coe-coe]` (where SC read = LDAR)

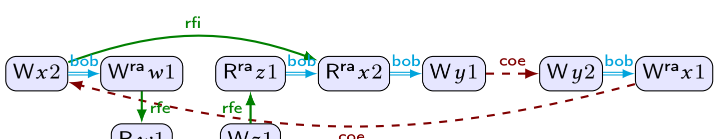
$$x := 2;\ s := z^{\mathsf{sc}};\ r := x^{\mathsf{sc}};\ y := 1 \parallel y := 2;\ x^{\mathsf{ra}} := 1 \qquad \text{(RFI-ACQ-COE-COE)}$$



(✓ARM8)

And release-acquire pairs `[rfi-ra-coe-coe]` (where acquiring read = LDAPR)

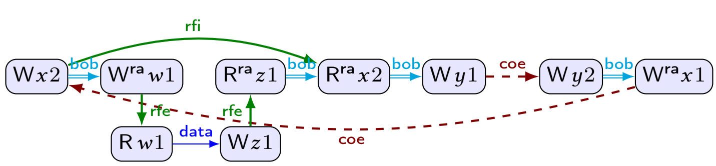
$$\begin{gathered} x := 2;\ w^{\mathsf{ra}} := 1;\ s := z^{\mathsf{ra}};\ r := x^{\mathsf{ra}};\ y := 1 \\ \parallel y := 2;\ x^{\mathsf{ra}} := 1 \parallel w := r;\ r := 1; \end{gathered} \qquad \text{(RFI-RA-COE-COE2)}$$



(✓ARM8)

But not if either acquire is strengthened to SC (where SC read = LDAR). The execution is also disallowed if an external thread places order between the ra accesses:

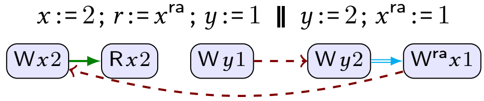
$$\begin{gathered} x := 2;\ w^{\mathsf{ra}} := 1;\ s := z^{\mathsf{ra}};\ r := x^{\mathsf{ra}};\ y := 1 \\ \parallel y := 2;\ x^{\mathsf{ra}} := 1 \parallel w := r;\ r := z; \end{gathered} \qquad \text{(RFI-RA-DATA-COE-COE)}$$



(✗ARM8)

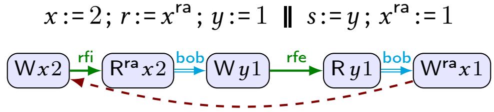
To allow this, weaken ra to rlx when read fulfilled by relaxed write of same thread (don't need to allow this when the write is part of an RMW).

$$x := 2;\ r := x^{\mathsf{ra}};\ y := 1 \parallel y := 2;\ x^{\mathsf{ra}} := 1$$



RF variant `[rfi-rfe-coe]`:

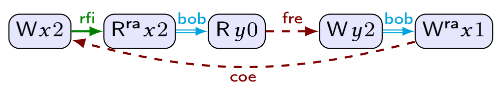
$$x := 2;\ r := x^{\mathsf{ra}};\ y := 1 \parallel s := y;\ x^{\mathsf{ra}} := 1 \qquad \text{(RFI-RFE-COE)}$$



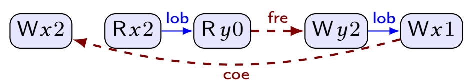
(✓ARM8)

TSO variant [rfi-fre-coe]:

$$x := 2;\ r := x^{\mathsf{ra}};\ s := y \ \parallel\ y := 2;\ x^{\mathsf{ra}} := 1 \qquad \text{(RFI-COE-COE)}$$



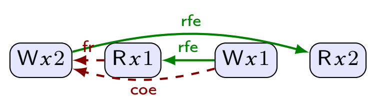
(✓ARM8)



(✓TSO)

Note that TSO does not order W to R in local order, even in poloc. Nonetheless, TSO disallows the following because of local visibility in first thread.

$$x := 2;\ r := x \ \parallel\ x := 1;\ s := x$$



(✗TSO)

[Higham and Kawash 2000] describe TSO as a linearization of partial order including:

- poloc
- lws = po; [W]
- $d \xrightarrow{\mathsf{po}} e$ when $c \xrightarrow{\mathsf{rfe}} d \xrightarrow{\mathsf{po}} e$

[Alglave et al. 2020] describe TSO as linearization of partial order satisfying internal visibility and including

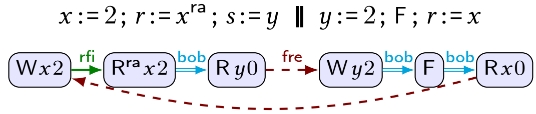
- [W]; po; [W]
- $d \xrightarrow{\mathsf{po}} e$ when $c \xrightarrow{\mathsf{rfe}} d \xrightarrow{\mathsf{po}} e$, from `(range(rfe) * _)`
- [R]; po; [W], from `(rfi^-1; lob)`

Ignoring fences and RMWs:

```
let rec lob = po \ ([W]; po; [R])
let IM0 = loc & ((IW * (M\IW)) | ((W\FW) * FW))
let gc-req = (W * _) | ((R * _) & ((range(rfe) * _) | (rfi^-1; lob))
let preorder-gcb = IM0 | lob & gc-req
```

Double FRE variant [rfi-fre-fre]:

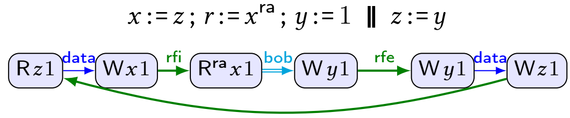
$$x := 2;\ r := x^{\mathsf{ra}};\ s := y \ \parallel\ y := 2;\ \mathsf{F};\ r := x \qquad \text{(RFI-FRE-FRE)}$$



(✓ARM8)

It does not seem possible to do this only with rfe. ARM disallows this [data-rfi-rfe-rfe]:

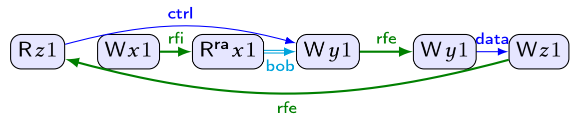
$$x := z;\ r := x^{\mathsf{ra}};\ y := 1 \ \parallel\ z := y \qquad \text{(DATA-RFI-RFE-RFE)}$$



(✗ARM8)

It also disallows [ctrl-rfi-rfe-rfe]:

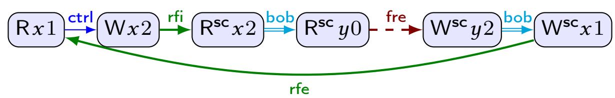
$$\texttt{if}(z)\{\};\ x:=1;\ r:=x^{\mathsf{ra}};\ y:=1\ \parallel\ z:=y \qquad (\textsc{ctrl-rfi-rfe-rfe})$$



(✗ARM8)

ARM allows some counterintuitive results for SC access [ctrl-rfi-fre-rfe]:

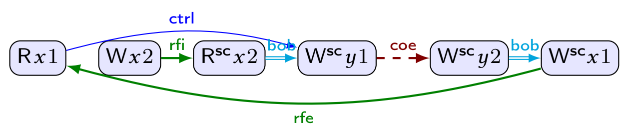
$$\texttt{if}(x)\{\};\ x:=2;\ r:=x^{\mathsf{sc}};\ s:=y^{\mathsf{sc}}\ \parallel\ y^{\mathsf{sc}}:=2;\ x^{\mathsf{sc}}:=1 \qquad (\textsc{ctrl-rfi-fre-rfe})$$



(✓ARM8)

Not possible with coe [ctrl-rfi-coe-rfe]:

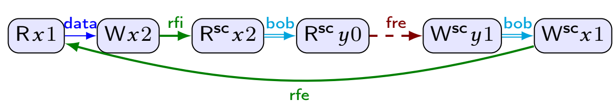
$$\texttt{if}(x)\{\};\ x:=2;\ r:=x^{\mathsf{sc}};\ y^{\mathsf{sc}}:=1\ \parallel\ y^{\mathsf{sc}}:=2;\ x^{\mathsf{sc}}:=1 \qquad (\textsc{ctrl-rfi-coe-rfe})$$



(✗ARM8)

This is not allowed with a data dependency instead of a control dependency [data-rfi-fre-rfe]:

$$x:=x{+}1;\ r:=x^{\mathsf{sc}};\ s:=y^{\mathsf{sc}}\ \parallel\ y^{\mathsf{sc}}:=1;\ x^{\mathsf{sc}}:=1 \qquad (\textsc{data-rfi-fre-rfe})$$

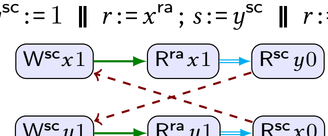


(✗ARM8)

7 SC EXAMPLES

IRIW-AQC-SC is allowed by trailing-sync compilation to power [Lahav et al. 2017, §1].

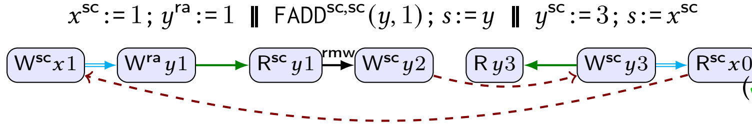
$$x^{\mathsf{sc}}:=1\ \parallel\ y^{\mathsf{sc}}:=1\ \parallel\ r:=x^{\mathsf{ra}};\ s:=y^{\mathsf{sc}}\ \parallel\ r:=y^{\mathsf{ra}};\ s:=x^{\mathsf{sc}} \qquad (\textsc{iriw-aqc-sc})$$



(✓POWER,RC11)

Leading sync is also unsound in C11 with RMW [Lahav et al. 2017, §2.1].

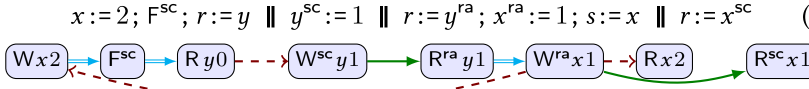
$$x^{\mathsf{sc}}:=1;\ y^{\mathsf{ra}}:=1\ \parallel\ \mathsf{FADD}^{\mathsf{sc,sc}}(y,1);\ s:=y\ \parallel\ y^{\mathsf{sc}}:=3;\ s:=x^{\mathsf{sc}} \qquad (\textsc{z6.u})$$



(✓POWER,RC11)

Leading sync is also unsound in C11 with SC fences [Lahav et al. 2017, §A.1].

$$x:=2;\ \mathsf{F}^{\mathsf{sc}};\ r:=y\ \parallel\ y^{\mathsf{sc}}:=1\ \parallel\ r:=y^{\mathsf{ra}};\ x^{\mathsf{ra}}:=1;\ s:=x\ \parallel\ r:=x^{\mathsf{sc}} \qquad (\textsc{rsync+rsc})$$

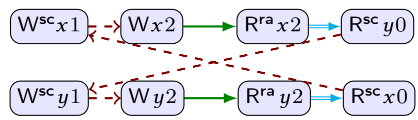


(✓RC11)

Fulfillment of $(\mathsf{R}\,x2)$ requires that either $(\mathsf{W}^{\mathsf{ra}}x1) \dashrightarrow (\mathsf{W}\,x2)$ or $(\mathsf{R}\,x2) \dashrightarrow (\mathsf{W}^{\mathsf{ra}}x1)$. It's interesting that in the pomset, $(\mathsf{R}^{\mathsf{sc}}x1)$ is not needed to get a cycle.

There is a long discussion of this in [Bender and Palsberg 2019, §5.2, Fig. 17], where they also discuss this example:

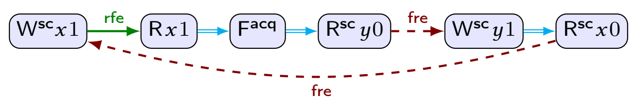
$$x^{\mathsf{sc}} := 1\,;\ x := 2 \parallel y^{\mathsf{sc}} := 1\,;\ y := 2 \parallel r := x^{\mathsf{ra}}\,;\ s := y^{\mathsf{sc}} \parallel r := y^{\mathsf{ra}}\,;\ s := x^{\mathsf{sc}} \qquad (\text{IRIW-SC-RLX-ACQ})$$



(✓RC11)

[Lahav et al. 2017, §A.2] claims that ARM8 allows this [RWC+acq+sc], but herd7 rejects it. Reason: they are citing the flowing/pop model [Flur et al. 2016] rather than [Pulte et al. 2018].

$$x^{\mathsf{sc}} := 1 \parallel r := x\,;\ \mathsf{F}^{\mathsf{acq}}\,;\ s := y^{\mathsf{sc}} \parallel y^{\mathsf{sc}} := 1\,;\ r := x^{\mathsf{sc}} \qquad (\text{RWC+ACQ+SC})$$



(✗ARM8)

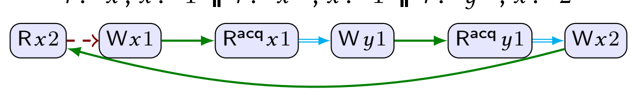
8 RMWS

From [Bender and Palsberg 2019, §3.3]. With partial coherence/weak fulfillment you need to be careful that RMWs are totally ordered (if that's a property you want). May not come for free.

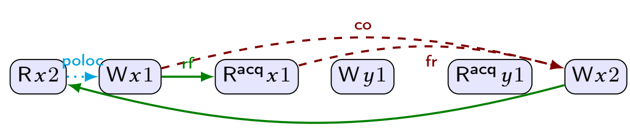
9 EXAMPLE FROM JAM PAPER

From [Bender and Palsberg 2019, §B]: "Here we demonstrate that it is possible to construct a program that is only forbidden due to the total coherence order"

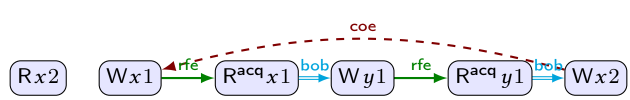
$$r := x\,;\ x := 1 \parallel r := x^{\mathsf{ra}}\,;\ x := 1 \parallel r := y^{\mathsf{ra}}\,;\ x := 2 \qquad (\text{TOTAL-CO})$$



(✗ARM8)



(✗ARM8)



(✗ARM8)

10 MORE MODEL

These definitions need to be updated to include the additional orders.

Definition 10.1. A pomset is x-*closed* if

- every $\mathcal{A}(e) = (\mathsf{R}x..)$ is fulfilled
- every $\Phi(e)$ is independent of x: $\big(\forall v.\ \Phi(e) \vDash \Phi(e)[v/x] \vDash \Phi(e)\big)$

Definition 10.2. Let $P \in (\nu x.\mathcal{P})$ when $P \in \mathcal{P}$ and P is x-closed

Definition 10.3. Let $P \in (\phi \rhd \mathcal{P})$ when $P \in \mathcal{P}$ and $(\forall e \in E)\ \Phi(e)$ implies ϕ

Definition 10.4. Let $P' \in (\mathcal{P}[M/x])$ when $(\exists P \in \mathcal{P})$
$E' = E$, $\leq' = \leq$, $\mathcal{A}' = \mathcal{A}$, and $(\forall e \in E')\ \Phi'(e) = \Phi(e)[M/x]$

Definition 10.5. Let $P' \in (\mathcal{P}^1 \parallel \mathcal{P}^2)$ when $(\exists P^1 \in \mathcal{P}^1)$ $(\exists P^2 \in \mathcal{P}^2)$ $E' = E^1 \cup E^2$, $\leq' \supseteq \leq^1 \cup \leq^2$, and $(\forall e \in E')$ either

$$e \notin E^2,\ \mathcal{A}'(e) = \mathcal{A}^1(e) \text{ and } \Phi'(e) \text{ implies } \Phi^1(e),$$
$$e \notin E^1,\ \mathcal{A}'(e) = \mathcal{A}^2(e) \text{ and } \Phi'(e) \text{ implies } \Phi^2(e),\ \text{ or}$$
$$\mathcal{A}'(e) = \mathcal{A}^1(e) = \mathcal{A}^2(e) \text{ and } \Phi'(e) \text{ implies } \Phi^1(e) \vee \Phi^2(e)$$

Language

$$\begin{aligned}
[\![\texttt{skip}]\!] &\triangleq \{\checkmark\}\\
[\![r := M\,;\, C]\!] &\triangleq [\![C]\!]\,[M/r]\\
[\![r := x^{\mu}\,;\, C]\!] &\triangleq \textstyle\bigcup_v\ (\mathsf{R}^{\mu} x v) \Rightarrow [\![C]\!]\,[x/r]\\
[\![x^{\mu} := M\,;\, C]\!] &\triangleq \textstyle\bigcup_v\ (M = v \mid \mathsf{W}^{\mu} x v) \Rightarrow [\![C]\!]\,[M/x]\\
[\![\mathsf{F}^{\nu}\,;\, C]\!] &\triangleq (\mathsf{F}^{\nu}) \Rightarrow [\![C]\!]\\
[\![\texttt{if}(M)\{C\}\,\texttt{else}\,\{D\}]\!] &\triangleq \big(M \rhd [\![C]\!]\big) \parallel \big(\neg M \rhd [\![D]\!]\big)\\
[\![C \parallel D]\!] &\triangleq [\![C]\!] \parallel [\![D]\!]\\
[\![\texttt{var}\, x\,;\, C]\!] &\triangleq \nu x\,.\,[\![C]\!]
\end{aligned}$$

$$\begin{aligned}
\nu \ ::=\ & \mathsf{rel} && \text{(Release)}\\
\mid\ & \mathsf{acq} && \text{(Acquire)}\\
\mid\ & \mathsf{sc} && \text{(SC)}
\end{aligned}$$

A citation: [Jagadeesan et al. 2020]

REFERENCES

Jade Alglave, Will Deacon, Richard Grisenthwaite, Antoine Hacquard, and Luc Maranget. 2020. Armed cats: Formal Concurrency Modelling at Arm. Draft. , 49 pages.

John Bender and Jens Palsberg. 2019. A formalization of Java's concurrent access modes. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 142:1–142:28. https://doi.org/10.1145/3360568

Shaked Flur, Kathryn E. Gray, Christopher Pulte, Susmit Sarkar, Ali Sezgin, Luc Maranget, Will Deacon, and Peter Sewell. 2016. Modelling the ARMv8 architecture, operationally: concurrency and ISA. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 608–621. https://doi.org/10.1145/2837614.2837615

Lisa Higham and Jalal Kawash. 2000. Memory Consistency and Process Coordination for SPARC Multiprocessors. In *High Performance Computing - HiPC 2000, 7th International Conference, Bangalore, India, December 17-20, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1970)*, Mateo Valero, Viktor K. Prasanna, and Sriram Vajapeyam (Eds.). Springer, 355–366. https://doi.org/10.1007/3-540-44467-X_32

Radha Jagadeesan, Alan Jeffrey, and James Riely. 2020. Pomsets with Preconditions: A Simple Model of Relaxed Memory. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 194 (Nov. 2020), 30 pages. https://doi.org/10.1145/3428262

Ori Lahav, Viktor Vafeiadis, Jeehoon Kang, Chung-Kil Hur, and Derek Dreyer. 2017. Repairing sequential consistency in C/C++11. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017*, Albert Cohen and Martin T. Vechev (Eds.). ACM, 618–632. https://doi.org/10.1145/3062341.3062352

Daniel Lustig, Sameer Sahasrabuddhe, and Olivier Giroux. 2019. A Formal Analysis of the NVIDIA PTX Memory Consistency Model. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, Iris Bahar, Maurice Herlihy, Emmett Witchel, and Alvin R. Lebeck (Eds.). ACM, 257–270. https://doi.org/10.1145/3297858.3304043

NVIDIA. 2020. Parallel Thread Execution ISA Version 7.1. https://docs.nvidia.com/cuda/parallel-thread-execution/index.html#memory-consistency-model.

Christopher Pulte, Shaked Flur, Will Deacon, Jon French, Susmit Sarkar, and Peter Sewell. 2018. Simplifying ARM concurrency: multicopy-atomic axiomatic and operational models for ARMv8. *PACMPL* 2, POPL (2018), 19:1–19:29. https://doi.org/10.1145/3158107