Program logics for weak memory concurrency

Viktor Vafeiadis

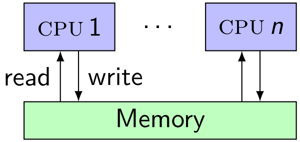
Max Planck Institute for Software Systems (MPI-SWS)

EPIT 2018
May 2018

The illusion of sequential consistency

Sequential consistency (SC)

- The standard simplistic concurrency model.
- Threads access shared memory in an interleaved fashion.



But. . .

- No multicore processor implements SC.
- Compiler optimisations invalidate SC.

Observable weak behaviour

Store buffering (SB)

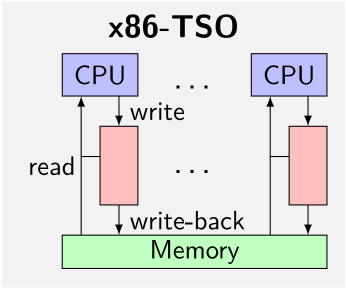
Initially, $x = y = 0$

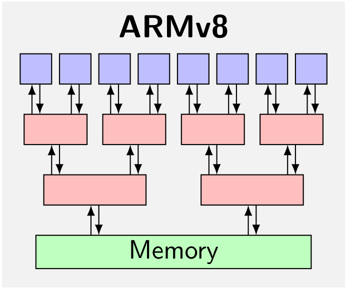
$x := 1;$	$y := 1;$
$a := y$ *// 0*	$b := x$ *// 0*

Load buffering (LB)

Initially, $x = y = 0$

$a := y;$ *// 1*	$b := x;$ *// 1*
$x := 1$	$y := 1$





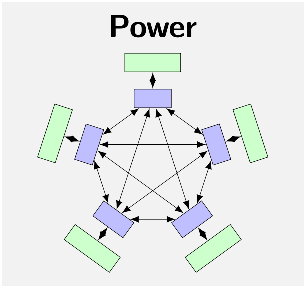
There is more to WMC than just reorderings [FM'16]

Independent reads of independent writes (IRIW)

Initially, $x = y = 0$

$$x := 1 \;\Bigg\|\; \begin{array}{l} a := x; \quad \textit{// 1} \\ \text{lwsync}; \\ b := y \quad \textit{// 0} \end{array} \;\Bigg\|\; \begin{array}{l} c := y; \quad \textit{// 1} \\ \text{lwsync}; \\ d := x \quad \textit{// 0} \end{array} \;\Bigg\|\; y := 1$$

- Thread II and III can observe the $x := 1$ and $y := 1$ writes happen in different orders.
- Because of the lwsync fences, no reorderings are possible!



Owicki-Gries method (1976)

OG = Hoare logic + rule for parallel composition

$$\frac{\left\{P_1\right\}\ c_1\ \left\{Q_1\right\} \qquad \left\{P_2\right\}\ c_2\ \left\{Q_2\right\} \quad \text{the two proofs are } \textit{non-interfering}}{\left\{P_1 \wedge P_2\right\}\ c_1 \parallel c_2\ \left\{Q_1 \wedge Q_2\right\}}$$

Non-interference

$R \wedge P \vdash R\{u/x\}$ for every:

- assertion R in the proof outline of one thread
- assignment $x := u$ with precondition P in the proof outline of the other thread

Standard OG is unsound for WMC [ICALP'15]

$$\{a \neq 0\}$$

$$\begin{array}{l|l} x := 1; & y := 1; \\ a := y & b := x \end{array}$$

$$\{a \neq 0 \lor b \neq 0\}$$

Standard OG is unsound for WMC

[ICALP'15]

$$\left\{a \neq 0\right\}$$

$$\begin{array}{l|l}
\left\{a \neq 0\right\} & \\
x := 1; & y := 1; \\
 & \\
a := y & b := x
\end{array}$$

$$\left\{a \neq 0 \vee b \neq 0\right\}$$

Standard OG is unsound for WMC

[ICALP'15]

$$\begin{array}{c}
\{a \neq 0\} \\
\begin{array}{l c l}
\{a \neq 0\} & & \\
x := 1; & \Bigg\| & y := 1; \\
\{x \neq 0\} & & \\
a := y & & b := x
\end{array} \\
\{a \neq 0 \lor b \neq 0\}
\end{array}$$

Standard OG is unsound for WMC [ICALP'15]

$$\{a \neq 0\}$$

$$\begin{array}{l|l}
\{a \neq 0\} & \\
x := 1; & y := 1; \\
\{x \neq 0\} & \\
a := y & b := x \\
\{x \neq 0\} &
\end{array}$$

$$\{a \neq 0 \vee b \neq 0\}$$

Standard OG is unsound for WMC [ICALP'15]

$$
\{a \neq 0\}
$$

$$
\left.\begin{array}{l}
\{a \neq 0\} \\
x := 1; \\
\{x \neq 0\} \\
a := y \\
\{x \neq 0\}
\end{array}\right\|
\begin{array}{l}
\{\top\} \\
y := 1; \\
\\
b := x \\
\\
\end{array}
$$

$$
\{a \neq 0 \vee b \neq 0\}
$$

Standard OG is unsound for WMC

[ICALP'15]

$$\{a \neq 0\}$$

$$
\begin{array}{l||l}
\{a \neq 0\} & \{\top\} \\
x := 1; & y := 1; \\
\{x \neq 0\} & \{y \neq 0\} \\
a := y & b := x \\
\{x \neq 0\} &
\end{array}
$$

$$\{a \neq 0 \lor b \neq 0\}$$

Standard OG is unsound for WMC [ICALP'15]

$$
\begin{array}{c}
\{a \neq 0\} \\
\begin{array}{l|l}
\{a \neq 0\} & \{\top\} \\
x := 1; & y := 1; \\
\{x \neq 0\} & \{y \neq 0\} \\
a := y & b := x \\
\{x \neq 0\} & \{y \neq 0 \land (a \neq 0 \lor b = x)\}
\end{array} \\
\{a \neq 0 \lor b \neq 0\}
\end{array}
$$

Standard OG is unsound for WMC [ICALP'15]

$$\{a \neq 0\}$$

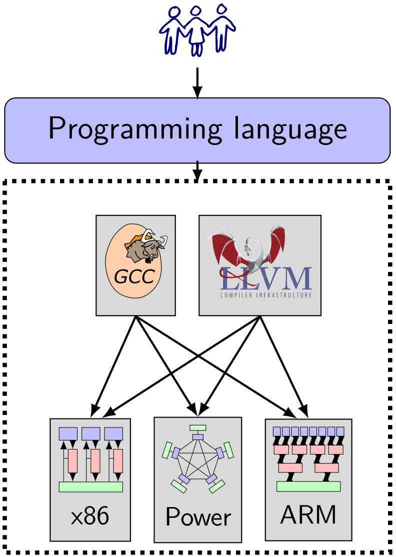
$$\begin{array}{l||l}
\{a \neq 0\} & \{\top\} \\
x := 1; & y := 1; \\
\{x \neq 0\} & \{y \neq 0\} \\
a := y & b := x \\
\{x \neq 0\} & \{y \neq 0 \wedge (a \neq 0 \vee b = x)\}
\end{array}$$

$$\{a \neq 0 \vee b \neq 0\}$$

Regaining soundness. . .

- Strengthen the non-inference check.
- OGRA: Owicki-Gries for release-acquire.

Which memory model?



Choose a PL model

- Platform-independence
- Takes into account the compiler optimisations

C/C++11

- The main existing model
- Many interesting features
- But also partially broken
- Use fixed version(s)

The C11 Memory Model

- Introduced in the C/C++ 2011 standards
- Formalized along with the standard [Batty et al., POPL'11]
- Many proposed fixes [OOPSLA'13, POPL'15, PLDI'17]

The C11 memory model: Atomics

Two types of locations

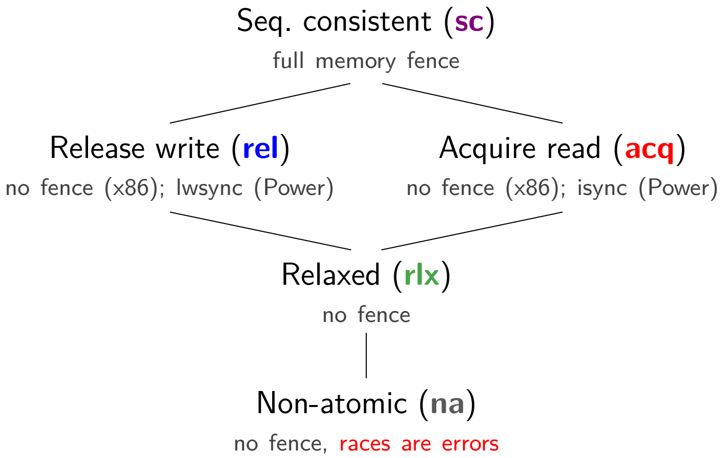
Ordinary (Non-Atomic)

Races are errors

Atomic

Welcome to the expert mode

A spectrum of accesses



Explicit primitives for fences

Store buffering in C11

Initially $x = y = 0$.

$$\begin{array}{l||l} x_{\mathsf{rlx}} := 1; & y_{\mathsf{rlx}} := 1; \\ a := y_{\mathsf{rlx}} \quad /\!/\, 0 & b := x_{\mathsf{rlx}} \quad /\!/\, 0 \end{array}$$

Can return $a = b = 0$ with the following execution:

$$\begin{tikzcd}
& {[x = y = 0]} \arrow[dl] \arrow[dr] \arrow[ddl, dashed] \arrow[ddr, dashed, "\mathtt{rf}\ \text{(reads-from)}"'] & \\
\mathtt{W}_{\mathsf{rlx}}\, x, 1 \arrow[d] & & \mathtt{W}_{\mathsf{rlx}}\, y, 1 \arrow[d, "\mathtt{po}\ \text{(program-order)}"] \\
\mathtt{R}_{\mathsf{rlx}}\, y, 0 & & \mathtt{R}_{\mathsf{rlx}}\, x, 0
\end{tikzcd}$$

Release-acquire synchronization

Initially $x = y = 0$.

$$\begin{array}{l||l}
 & \textbf{repeat} \\
x_{\text{na}} := 5; & \quad a := y_{\text{acq}}; \\
y_{\text{rel}} := 1 & \textbf{until } a \neq 0; \\
 & b := x_{\text{na}}
\end{array}$$

One possible execution:

$$\begin{tikzcd}
 & {[x = y = 0]} \arrow[dl] \arrow[dr] \arrow[dr, dashed, bend left, "\texttt{rf} \text{ (reads-from)}"] & \\
\mathtt{W}_{\text{na}}\ x, 5 \arrow[d] \arrow[drr, dashed] & & \mathtt{R}_{\text{acq}}\ y, 0 \arrow[d, "\text{po (program-order)}"] \\
\mathtt{W}_{\text{rel}}\ y, 1 \arrow[rr, bend left, "\overset{\text{(sync.with)}}{\texttt{sw}}"] \arrow[rr, dashed, "\texttt{rf}"'] & & \mathtt{R}_{\text{acq}}\ y, 1 \arrow[d] \\
 & & \mathtt{R}_{\text{na}}\ x, 5
\end{tikzcd}$$

$\texttt{hb} \triangleq (\text{po} \cup \texttt{sw})^+$

Coherence

Programs with a single shared variable behave as under SC.

$$\begin{array}{l||l} x_{\mathbf{rlx}} := 1; & a := x_{\mathbf{rlx}}; \quad /\!/ \, 2 \\ x_{\mathbf{rlx}} := 2 & b := x_{\mathbf{rlx}} \quad /\!/ \, 1 \end{array}$$

The outcome $a = 2 \wedge b = 1$ is forbidden.

$$\begin{tikzcd}
\mathtt{W}_{\mathbf{rlx}}\, x, 1 \arrow[d, bend right, "\mathtt{mo}_x"'] \arrow[d] \arrow[dr, dashed] & \mathtt{R}_{\mathbf{rlx}}\, x, 2 \arrow[d] \\
\mathtt{W}_{\mathbf{rlx}}\, x, 2 \arrow[ur, dashed] & \mathtt{R}_{\mathbf{rlx}}\, x, 1 \arrow[l, dotted, "\mathtt{rb}_x"]
\end{tikzcd}$$

- Modification order, $\mathtt{mo}_x$, total order of writes to x.
- Reads-before : $\mathtt{rb} \triangleq (\mathtt{rf}^{-1}; \mathtt{mo}) \cap (\neq)$
- Coherence : $\mathtt{hb} \cup \mathtt{rf}_x \cup \mathtt{mo}_x \cup \mathtt{rb}_x$ is acyclic for all x.

Relaxed program logics

- RSL (relaxed separation logic, OOPSLA'13)
- FSL (fenced separation logic, VMCAI'16)
- GPS (ghosts & protocols, OOPSLA'14, PLDI'15)

Separation logic

Key concept of *ownership* :

- Resourceful reading of Hoare triples.

$$\{P\}\ C\ \{Q\}$$

- To access a non-atomic location, you must own it:

$$\begin{array}{lll}
\{\mathrm{emp}\} & a := \textbf{alloc} & \{a \mapsto \_\} \\
\{x \mapsto v\} & a := x_{\mathbf{na}} & \{x \mapsto v \wedge a = v\} \\
\{x \mapsto v\} & x_{\mathbf{na}} := v' & \{x \mapsto v'\}
\end{array}$$

- Disjoint parallelism:

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \qquad \{P_2\}\ C_2\ \{Q_2\}}{\{P_1 * P_2\}\ C_1 \| C_2\ \{Q_1 * Q_2\}}$$

Separation logic: Disjoint parallelism

$$
\begin{array}{c}
\left\{x \mapsto 0 * y \mapsto 0\right\} \\
\begin{array}{l||l}
\left\{x \mapsto 0\right\} & \left\{y \mapsto 0\right\} \\
a := x_{\mathbf{na}}; & b := y_{\mathbf{na}}; \\
\left\{x \mapsto 0 \land a = 0\right\} & \left\{y \mapsto 0 \land b = 0\right\} \\
x_{\mathbf{na}} := a + 1; & y_{\mathbf{na}} := b + 1; \\
\left\{x \mapsto 1\right\} & \left\{y \mapsto 1\right\}
\end{array} \\
\left\{x \mapsto 1 * y \mapsto 1\right\}
\end{array}
$$

Simple programs are easy to verify!

Rules for release/acquire accesses [OOPSLA'13]

Ownership transfer by release/acquire synchronizations.

- Initially, pick location invariant $\mathcal{Q}$.

$$x \mapsto v * \mathcal{Q}(v) \Rrightarrow \mathbf{W}_{\mathcal{Q}}(x) * \mathbf{R}_{\mathcal{Q}}(x)$$

- Release write $\leadsto$ give away permissions.

$$\left\{\mathbf{W}_{\mathcal{Q}}(x) * \mathcal{Q}(v)\right\}\ x_{\mathbf{rel}} := v\ \left\{\mathbf{W}_{\mathcal{Q}}(x)\right\}$$

- Acquire read $\leadsto$ gain permissions.

$$\left\{\mathbf{R}_{\mathcal{Q}}(x)\right\}\ a := x_{\mathbf{acq}}\ \left\{\mathbf{R}_{\mathcal{Q}[a:=\mathrm{emp}]}(x) * \mathcal{Q}(a)\right\}$$

where $\mathcal{Q}[a{:=}\mathrm{emp}] \triangleq \lambda v.\ \mathbf{if}\ v = a\ \mathbf{then}\ \mathrm{emp}\ \mathbf{else}\ \mathcal{Q}(v)$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \vee x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\begin{array}{l||l} x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\ y_{\mathbf{rel}} := 1; & \\ & \textbf{if } a \neq 0 \textbf{ then } b := x_{\mathbf{na}} \end{array}$$

$$\left\{a = 0 \vee b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\right\}$$

$$\begin{array}{l|l} x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\ y_{\mathbf{rel}} := 1; & \\ & \mathbf{if}\ a \neq 0\ \mathbf{then}\ b := x_{\mathbf{na}} \end{array}$$

$$\left\{a = 0 \lor b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \vee x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\right\}$$

```
{x ↦ 0 * W_Q(y)}     ||  {R_Q(y)}
x_na := 5;           ||  a := y_acq
                     ||
y_rel := 1;          ||
                     ||  if a ≠ 0 then b := x_na
```

$$\left\{a = 0 \vee b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$
$$\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\right\}$$

$$\begin{array}{l||l}
\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{\mathbf{R}_{\mathcal{Q}}(y)\right\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\left\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \\
y_{\mathbf{rel}} := 1; & \\
 & \mathbf{if}\ a \neq 0\ \mathbf{then}\ b := x_{\mathbf{na}}
\end{array}$$

$$\left\{a = 0 \lor b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\{x \mapsto 0 * y \mapsto 0\}$$

$$\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\}$$

$$\left.\begin{array}{l} \{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\} \\ x_{\mathbf{na}} := 5; \\ \{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\} \\ y_{\mathbf{rel}} := 1; \\ \{\mathbf{W}_{\mathcal{Q}}(y)\} \end{array}\right\| \begin{array}{l} \{\mathbf{R}_{\mathcal{Q}}(y)\} \\ a := y_{\mathbf{acq}} \\ \\ \\ \mathbf{if}\ a \neq 0\ \mathbf{then}\ b := x_{\mathbf{na}} \end{array}$$

$$\{a = 0 \lor b = 5\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \vee x \mapsto 5)$.

$$\{x \mapsto 0 * y \mapsto 0\}$$
$$\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\}$$
$$\begin{array}{l||l}
\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\} & \{\mathbf{R}_{\mathcal{Q}}(y)\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\} & \\
y_{\mathbf{rel}} := 1; & \\
\{\mathbf{W}_{\mathcal{Q}}(y)\} & \textbf{if } a \neq 0 \textbf{ then } b := x_{\mathbf{na}} \\
\{\top\} &
\end{array}$$
$$\{a = 0 \vee b = 5\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \vee x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\right\}$$

$$\begin{array}{l||l}
\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{\mathbf{R}_{\mathcal{Q}}(y)\right\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\left\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{(a = 0 \vee x \mapsto 5) * \mathbf{R}_{\mathcal{Q}[a:=\mathsf{emp}]}(y)\right\} \\
y_{\mathbf{rel}} := 1; & \\
\left\{\mathbf{W}_{\mathcal{Q}}(y)\right\} & \textbf{if } a \neq 0 \textbf{ then } b := x_{\mathbf{na}} \\
\left\{\top\right\} &
\end{array}$$

$$\left\{a = 0 \vee b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\{x \mapsto 0 * y \mapsto 0\}$$
$$\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\}$$
$$\begin{array}{l||l}
\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\} & \{\mathbf{R}_{\mathcal{Q}}(y)\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\} & \{(a = 0 \lor x \mapsto 5) * \mathbf{R}_{\mathcal{Q}[a:=\mathsf{emp}]}(y)\} \\
y_{\mathbf{rel}} := 1; & \{a = 0 \lor x \mapsto 5\} \\
\{\mathbf{W}_{\mathcal{Q}}(y)\} & \mathbf{if}\ a \neq 0\ \mathbf{then}\ b := x_{\mathbf{na}} \\
\{\top\} &
\end{array}$$
$$\{a = 0 \lor b = 5\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\right\}$$

$$\begin{array}{l||l}
\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{\mathbf{R}_{\mathcal{Q}}(y)\right\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\left\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{(a = 0 \lor x \mapsto 5) * \mathbf{R}_{\mathcal{Q}[a:=\mathrm{emp}]}(y)\right\} \\
y_{\mathbf{rel}} := 1; & \left\{a = 0 \lor x \mapsto 5\right\} \\
\left\{\mathbf{W}_{\mathcal{Q}}(y)\right\} & \mathbf{if}\ a \neq 0\ \mathbf{then}\ b := x_{\mathbf{na}} \\
\left\{\top\right\} & \left\{a = 0 \lor (x \mapsto 5 \land b = 5)\right\}
\end{array}$$

$$\left\{a = 0 \lor b = 5\right\}$$

Release-acquire synchronization: message passing

Let $\mathcal{Q}(v) \triangleq (v = 0 \lor x \mapsto 5)$.

$$\{x \mapsto 0 * y \mapsto 0\}$$
$$\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y) * \mathbf{R}_{\mathcal{Q}}(y)\}$$
$$\begin{array}{l||l}
\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\} & \{\mathbf{R}_{\mathcal{Q}}(y)\} \\
x_{\mathbf{na}} := 5; & a := y_{\mathbf{acq}} \\
\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\} & \{(a = 0 \lor x \mapsto 5) * \mathbf{R}_{\mathcal{Q}[a:=\mathsf{emp}]}(y)\} \\
y_{\mathbf{rel}} := 1; & \{a = 0 \lor x \mapsto 5\} \\
\{\mathbf{W}_{\mathcal{Q}}(y)\} & \textbf{if } a \neq 0 \textbf{ then } b := x_{\mathbf{na}} \\
\{\top\} & \{a = 0 \lor (x \mapsto 5 \land b = 5)\}
\end{array}$$
$$\{a = 0 \lor b = 5\}$$

Ownership transfer works!

Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

$$\left\{\mathbf{R}_{\mathcal{Q}}(x)\right\}\ a := x_{\mathsf{rlx}}\ \left\{\mathbf{R}_{\mathcal{Q}}(x) \wedge (\mathcal{Q}(a) \not\equiv \mathit{false})\right\}$$

- Relaxed writes:

$$\frac{\mathcal{Q}(v) = \mathsf{emp}}{\left\{\mathbf{W}_{\mathcal{Q}}(x)\right\}\ x_{\mathsf{rlx}} := v\ \left\{\mathbf{W}_{\mathcal{Q}}(x)\right\}}$$

Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

$$\left\{\mathbf{R}_{\mathcal{Q}}(x)\right\}\ a := x_{\mathsf{rlx}}\ \left\{\mathbf{R}_{\mathcal{Q}}(x) \land (\mathcal{Q}(a) \not\equiv \mathit{false})\right\}$$

- Relaxed writes:

$$\frac{\mathcal{Q}(v) = \mathsf{emp}}{\left\{\mathbf{W}_{\mathcal{Q}}(x)\right\}\ x_{\mathsf{rlx}} := v\ \left\{\mathbf{W}_{\mathcal{Q}}(x)\right\}}$$

Unsound because of dependency cycles!

Dependency cycles

Initially $x = y = 0$.

$$\begin{array}{l||l} a := x_{\mathbf{rlx}}; & b := y_{\mathbf{rlx}}; \\ \textbf{if } a \neq 0 \textbf{ then} & \textbf{if } b \neq 0 \textbf{ then} \\ \qquad y_{\mathbf{rlx}} := 1 & \qquad x_{\mathbf{rlx}} := 1 \end{array}$$

C11 allows the outcome $x = y = 1$.

Justification

$$\begin{tikzcd}
\mathtt{R}_{\mathbf{rlx}}\, x, 1 \arrow[d] & \mathtt{R}_{\mathbf{rlx}}\, y, 1 \arrow[d] \\
\mathtt{W}_{\mathbf{rlx}}\, y, 1 \arrow[ur, dashed] & \mathtt{W}_{\mathbf{rlx}}\, x, 1 \arrow[ul, dashed]
\end{tikzcd}$$

Relaxed accesses don't synchronize

Dependency cycles

Initially $x = y = 0$.

$$\begin{array}{l||l} a := x_{\textbf{rlx}}; & b := y_{\textbf{rlx}}; \\ \textbf{if } a \neq 0 \textbf{ then} & \textbf{if } b \neq 0 \textbf{ then} \\ \quad y_{\textbf{rlx}} := 1 & \quad x_{\textbf{rlx}} := 1 \end{array}$$

C11 allows the outcome $x = y = 1$.

What goes wrong:

Non-relational invariants are unsound.

$$x = 0 \wedge y = 0$$

The DRF-property does not hold.

Dependency cycles

Initially $x = y = 0$.

$$\begin{array}{l|l} a := x_{\mathsf{rlx}}; & b := y_{\mathsf{rlx}}; \\ \textbf{if } a \neq 0 \textbf{ then} & \textbf{if } b \neq 0 \textbf{ then} \\ \quad y_{\mathsf{rlx}} := 1 & \quad x_{\mathsf{rlx}} := 1 \end{array}$$

C11 allows the outcome $x = y = 1$.

A simple fix:
Strengthen the model to forbid $\mathtt{po} \cup \mathtt{rf}$ cycles.

A better fix:
Use the "promising" model [Kang et al., POPL'17]

Incorrect message passing

Initially $x = y = 0$.

$$
\begin{array}{l||l}
 & \textbf{repeat} \\
x_{\textbf{na}} := 5; & \quad a := y_{\textbf{rlx}} \\
 & \textbf{until } a \neq 0; \\
y_{\textbf{rlx}} := 1 & \\
 & b := x_{\textbf{na}}
\end{array}
$$

$$\begin{tikzcd}
\mathtt{W}_{\textbf{na}}\, x, 5 \arrow[d] \arrow[dr, dash, "\text{race}"] & \mathtt{R}_{\textbf{rlx}}\, y, 1 \arrow[d] \\
\mathtt{W}_{\textbf{rlx}}\, y, 1 \arrow[ur, dashed] & \mathtt{R}_{\textbf{na}}\, x, \textbf{?}
\end{tikzcd}$$

Message passing with C11 memory fences

Initially $x = y = 0$.

$$\begin{array}{l||l}
 & \textbf{repeat} \\
x_{\textsf{na}} := 5; & \quad a := y_{\textsf{rlx}} \\
\textbf{fence}(\textbf{rel}); & \textbf{until } a \neq 0; \\
y_{\textsf{rlx}} := 1 & \textbf{fence}(\textbf{acq}); \\
 & b := x_{\textsf{na}}
\end{array}$$

$$\begin{tikzcd}
\mathtt{W}_{\textsf{na}}\, x, 5 \arrow[d] \arrow[drr, dashed] & & \mathtt{R}_{\textsf{rlx}}\, y, 1 \arrow[d] \\
\mathtt{F}_{\textsf{rel}} \arrow[rr, "\mathtt{sw}"] \arrow[d] & & \mathtt{F}_{\textsf{acq}} \arrow[d] \\
\mathtt{W}_{\textsf{rlx}}\, y, 1 \arrow[uurr, dashed] & & \mathtt{R}_{\textsf{na}}\, x, 5
\end{tikzcd}$$

Fenced separation logic [VMCAI'16]

Introduce two 'modalities' in the logic:

- $\triangle P$: state ready to be transferred away.
- $\triangledown P$: state that will be acquired after a **fence**(**acq**).

Proof rules:

$$\{P\}\ \textbf{fence}(\textbf{rel})\ \{\triangle P\}$$

$$\{\mathbf{W}_{\mathcal{Q}}(x) * \triangle \mathcal{Q}(v)\}\quad x_{\mathbf{rlx}} := v \quad \{\mathbf{W}_{\mathcal{Q}}(x)\}$$

$$\{\mathbf{R}_{\mathcal{Q}}(x)\}\quad t := x_{\mathbf{rlx}} \quad \{\mathbf{R}_{\mathcal{Q}[t:=\mathrm{emp}]}(x) * \triangledown \mathcal{Q}(t)\}$$

$$\{\triangledown P\}\ \textbf{fence}(\textbf{acq})\ \{P\}$$

Message passing with C11 memory fences

Let $\mathcal{Q}(v) \triangleq v = 0 \lor x \mapsto 5$.

$$\left\{x \mapsto 0 * y \mapsto 0\right\}$$

$$\begin{array}{l|l}
 & \left\{\mathbf{R}_{\mathcal{Q}}(y)\right\} \\
 & a := y_{\mathbf{rlx}} \\
\left\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \left\{\triangledown(a = 0 \lor x \mapsto 5)\right\} \\
x_{\mathbf{na}} := 5; & \textbf{if } a \neq 0 \textbf{ then} \\
\left\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \qquad \left\{\triangledown(x \mapsto 5)\right\} \\
\textbf{fence}(\textbf{rel}); & \qquad \textbf{fence}(\textbf{acq}) \\
\left\{\triangle(x \mapsto 5) * \mathbf{W}_{\mathcal{Q}}(y)\right\} & \qquad \left\{x \mapsto 5\right\} \\
y_{\mathbf{rlx}} := 1; & \qquad b := x_{\mathbf{na}} \\
\left\{\mathbf{W}_{\mathcal{Q}}(y)\right\} & \qquad \left\{x \mapsto 5 \land b = 5\right\} \\
 & \left\{a = 0 \lor (x \mapsto 5 \land b = 5)\right\}
\end{array}$$

$$\left\{a = 0 \lor b = 5\right\}$$

GPS: A better logic for release-acquire [OOPSLA'14, PLDI'15]

Three key features:

- Location protocols
- Ghost state/tokens
- Escrows for ownership transfer

Example (Racy message passing)

Initially, $x = y = 0$.

$$\begin{array}{l} x_{\mathsf{rlx}} := 1; \\ y_{\mathsf{rel}} := 1 \end{array} \;\Bigg\|\; \begin{array}{l} x_{\mathsf{rlx}} := 1; \\ y_{\mathsf{rel}} := 1 \end{array} \;\Bigg\|\; \begin{array}{l} a := y_{\mathsf{acq}}; \\ b := x_{\mathsf{rlx}} \end{array}$$

Cannot get $a = 1 \wedge b = 0$.

Racy message passing in GPS

Protocol for x: $$\begin{tikzcd} \boxed{\textbf{A: } x = 0} \arrow[r] & \boxed{\textbf{B: } x = 1} \end{tikzcd}$$

Protocol for y: $$\begin{tikzcd} \boxed{\textbf{C: } y = 0} \arrow[r] & \boxed{\textbf{D: } y = 1 \land x.st \geq \textbf{B}} \end{tikzcd}$$

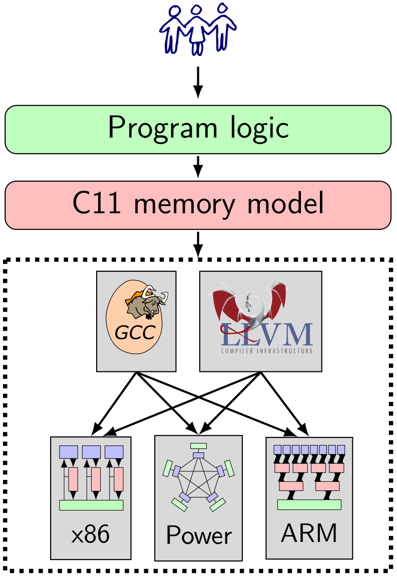
Acquire reads gain knowledge, not ownership.

$$
\left.\begin{array}{l}
\left\{x.st \geq \textbf{A} \land y.st \geq \textbf{C}\right\} \\
x_{\textbf{rlx}} := 1; \\
\left\{x.st \geq \textbf{B} \land y.st \geq \textbf{C}\right\} \\
y_{\textbf{rel}} := 1 \\
\left\{x.st \geq \textbf{B} \land y.st \geq \textbf{D}\right\}
\end{array}\right\|
\begin{array}{l}
\left\{x.st \geq \textbf{A} \land y.st \geq \textbf{C}\right\} \\
a := y_{\textbf{acq}}; \\
\left\{\begin{array}{l} a = 0 \land x.st \geq \textbf{A} \\ \lor\ a = 1 \land x.st \geq \textbf{B} \end{array}\right\} \\
b := x_{\textbf{rlx}}; \\
\left\{a = 0 \lor (a = 1 \land b = 1)\right\}
\end{array}
$$

Summary

Relaxed program logics

- RSL, FSL, GPS, . . .
- Reason about a strengthening of C11.
- Encodes common synchronisation patterns.
- Useful for explaning the weak memory model.


Program logic
C11 memory model
GCC
LLVM
COMPILER INFRASTRUCTURE
x86
Power
ARM