Relaxed program logics

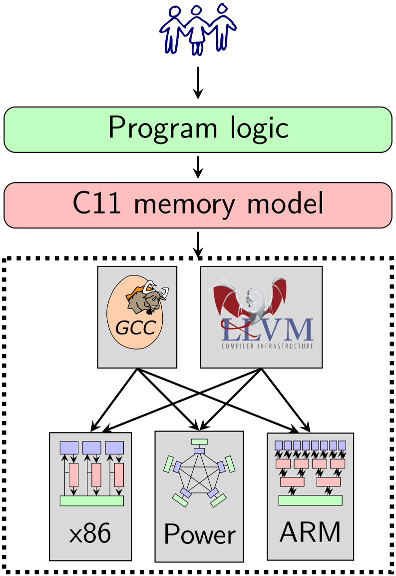
Ori Lahav Viktor Vafeiadis

1 September 2017

Overview

Relaxed program logics

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



Reminder: Separation logic

Key concept of *ownership* :

- Resourceful reading of Hoare triples.

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

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

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

- Disjoint parallelism:

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

Relaxed separation logic

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.

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

- Acquire read $\leadsto$ gain permissions.

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

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

Release-acquire synchronization: message passing

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

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

$x_{\mathbf{na}} := 5;$	$a := y_{\mathbf{acq}}$
$y_{\mathbf{rel}} := 1;$	$\textbf{if } a \neq 0 \textbf{ then } b := x_{\mathbf{na}}$

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

Release-acquire synchronization: message passing

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

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

$x_{\mathbf{na}} := 5;$	$a := y_{\mathbf{acq}}$
$y_{\mathbf{rel}} := 1;$	
	if $a \neq 0$ **then** $b := x_{\mathbf{na}}$

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

Release-acquire synchronization: message passing

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

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

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

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

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

Release-acquire synchronization: message passing

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

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

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

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

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

Release-acquire synchronization: message passing

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

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

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

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

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

Release-acquire synchronization: message passing

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

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

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

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

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

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

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

$\{x \mapsto 0 * \mathbf{W}_{\mathcal{Q}}(y)\}$	$\{\mathbf{R}_{\mathcal{Q}}(y)\}$
$x_{\mathbf{na}} := 5;$	$a := y_{\mathbf{acq}}$
$\{x \mapsto 5 * \mathbf{W}_{\mathcal{Q}}(y)\}$	$\{(a = 0 \vee x \mapsto 5) * \mathbf{R}_{\mathcal{Q}[a:=\mathrm{emp}]}(y)\}$
$y_{\mathbf{rel}} := 1;$	
$\{\mathbf{W}_{\mathcal{Q}}(y)\}$	**if** $a \neq 0$ **then** $b := x_{\mathbf{na}}$
$\{\top\}$	

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

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

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

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

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

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

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

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

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

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

Ownership transfer works!

Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

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

- Relaxed writes:

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

Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

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

- Relaxed writes:

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

Unsound because of dependency cycles!

Soundness statement

Definition (Memory safety)

An execution G is *memory safe* if every access in G “happens after” the allocation of the accessed location.

Definition (Data race)

G *has a data race* if there exist two hb-unrelated accesses in G to the same location such that
(a) at least one access is non-atomic, and
(b) at least one access is a write.

Theorem (Adequacy)

If $\{\text{true}\}$ *Prg* $\{\text{true}\}$*, then all consistent executions of Prg are memory safe and have no data races.*

Three technical challenges

Assertions in heaps

- Store syntactic assertions (modulo $*$-ACI)

No (global) notions of state and time

- Define a *logical* local notion of state
- Annotate `hb` edges with logical state

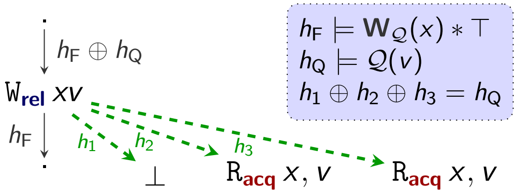
Declarative semantics

- Induct over max `hb`-path distance from top

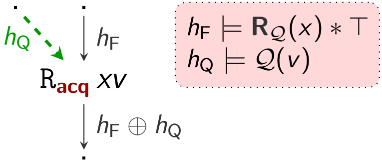
Local annotation validity

For node n roughly, $\sum_{e \in In(n)} hmap(e) + \text{effect}(n) \approx \sum_{e \in Out(n)} hmap(e)$

Release writes:



Acquire reads:



Independent heap compatibility

Definition (Pairwise independence)

A set $\mathcal{T}$ of edges is *pairwise independent* in a graph G iff $\forall (a, a'), (b, b') \in \mathcal{T},\ (a', b) \notin G.\mathtt{hb}^*$

Lemma (Independent heap compatibility)

If hmap is a locally valid annotation of execution graph G and $\mathcal{T} \subseteq G.\mathtt{hb}$ is pairwise independent in G, then $\bigoplus_{x \in \mathcal{T}} hmap(x)$ is defined.

Soundness

Configuration safety: A valid annotation can be extended for n further events.

Lemma (RSL triple $\Rightarrow$ annotation validity)

Let $\{\mathrm{true}\}\ c\ \{\mathrm{true}\}$. *Then, every consistent execution graph* $G \in [\![c]\!]$ *has a valid annotation.*

Theorem (Race-Freedom)

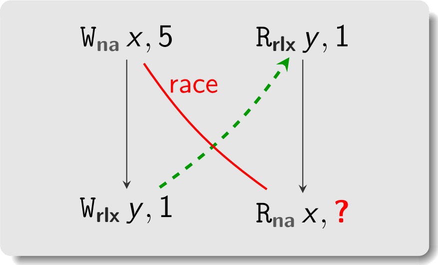
If $\{\mathrm{true}\}\ c\ \{\mathrm{true}\}$, $\forall G \in [\![c]\!]$, G *is race-free.*

Fenced separation logic

Incorrect message passing

Initially $x = y = 0$.

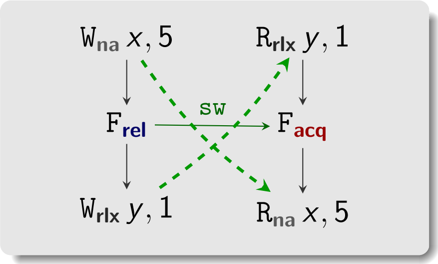
$x_{\textbf{na}} := 5;$ $y_{\textbf{rlx}} := 1$	**repeat** $\quad a := y_{\textbf{rlx}}$ **until** $a \neq 0;$ $b := x_{\textbf{na}}$



Message passing with C11 memory fences

Initially $x = y = 0$.

$x_{\mathsf{na}} := 5;$ **fence**(**rel**); $y_{\mathsf{rlx}} := 1$	**repeat** $a := y_{\mathsf{rlx}}$ **until** $a \neq 0;$ **fence**(**acq**); $b := x_{\mathsf{na}}$



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:

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

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

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

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

Message passing with C11 memory fences

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

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

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

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

GPS

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: $\textbf{A: } x = 0 \longrightarrow \textbf{B: } x = 1$

Protocol for y: $\textbf{C: } y = 0 \longrightarrow \textbf{D: } y = 1 \wedge x.st \geq \textbf{B}$

Acquire reads gain knowledge, not ownership.

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

GPS ghosts and escrows

To gain ownership, we use ghost state & escrows.

$$\frac{P * P \Rrightarrow \mathit{false}}{Q \Rrightarrow \mathbf{Esc}(P, Q)} \qquad \frac{}{\mathbf{Esc}(P, Q) * P \Rrightarrow Q}$$

Example (Message passing using escrows)

Invariant for x: $x = 0 \vee \mathbf{Esc}(K, \&a \mapsto 7)$.

$\{\&a \mapsto 0\}$	$\{K\}$
$a = 7;$	**if** $(x.\text{load}(acq) \neq 0)$
$\{\&a \mapsto 7\}$	$\quad \{K * \mathbf{Esc}(K, \&a \mapsto 7)\}$
$\{\mathbf{Esc}(K, \&a \mapsto 7)\}$	$\quad \{\&a \mapsto 7\}$
$x.\text{store}(1, rel);$	$\quad$ **print**(a);

SC fences

- An SC fence is roughly a release/acquire fence and a RMW to a distinguished location.

$$\frac{J * P * P' \Rrightarrow J * Q * Q'}{J \vdash \{P * \triangledown P'\}\ \mathbf{fence}(\mathbf{sc})\ \{Q * \vartriangle Q'\}}$$

SC accesses

- Program logics for SC $\rightsquigarrow$ multilocation invariants.
- What if SC and non-SC atomics are mixed?
  (C11 got the semantics wrong; see [PLDI'17].)
- Lack of useful programs to verify.

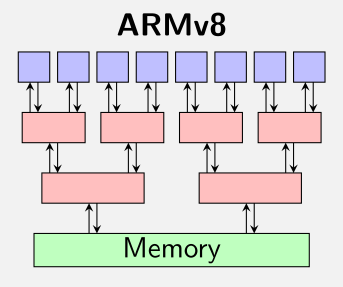
Challenge #2. Soundness under weaker memory models

- Soundness proofs require $\texttt{po} \cup \texttt{rf}$ acyclicity, which disallows the weak behaviour of LB.
- Even the logic is too strong.

Load buffering (LB)

Initially, $x = y = 0$

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



Towards a solution

- Promising semantics [Kang et al., POPL'17]
- iGPS [Kaiser et al., ECOOP'17]

Further reading

- Relaxed separation logic: A program logic for C11 concurrency. V. Vafeiadis, C. Narayan. OOPSLA 2013: 867-884
- GPS: Navigating weak memory with ghosts, protocols, and separation. A. Turon, V. Vafeiadis, Derek Dreyer. OOPSLA 2014: 691-707
- A program logic for C11 memory fences. M. Doko, V. Vafeiadis. VMCAI 2016: 413-430
- Tackling real-life relaxed concurrency with FSL++. M. Doko, V. Vafeiadis. ESOP 2017: 448-475
- Strong logic for weak memory: Reasoning about release-acquire consistency in Iris. J.-O. Kaiser, H.-H. Dang, D. Dreyer, O. Lahav, V. Vafeiadis. ECOOP 2017: 17:1-17:29