Relaxed program logics

Ori Lahav    Viktor Vafeiadis

1 September 2017
Relaxed program logics

- RSL, FSL, GPS, ... 
- Reason about a strengthening of C11.
- Encodes common synchronisation patterns.
- Useful for explaining 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:
  \[
  \{\text{emp}\} \ a := \text{alloc} \quad \{a \mapsto \_\} \\
  \{x \mapsto v\} 
  a := x_{\text{na}} \quad \{x \mapsto v \land a = v\} \\
  \{x \mapsto v\} \ x_{\text{na}} := v' \quad \{x \mapsto v'\}
  \]

- Disjoint parallelism:
  \[
  \{P_1\} \ C_1 \ \{Q_1\} \quad \{P_2\} \ C_2 \ \{Q_2\} \\
  \{P_1 \ast P_2\} \ C_1 \parallel C_2 \ \{Q_1 \ast Q_2\}
  \]
Relaxed separation logic
Ownership transfer by release/acquire synchronizations.

- Initially, pick location invariant $Q$.

\[ x \mapsto v \ast Q(v) \Implies W_Q(x) \ast R_Q(x) \]

- Release write $\rightsquigarrow$ give away permissions.

\[ \{ W_Q(x) \ast Q(v) \} \ x_{rel} \coloneqq v \ \{ W_Q(x) \} \]

- Acquire read $\rightsquigarrow$ gain permissions.

\[ \{ R_Q(x) \} \ a \coloneqq x_{acq} \ \{ R_{Q[a:=\text{emp}]}(x) \ast Q(a) \} \]

where $Q[a:=\text{emp}] \triangleq \lambda v. \ \text{if } v = a \ \text{then } \text{emp} \ \text{else } Q(v)$
Release-acquire synchronization: message passing

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

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

$x_{na} := 5$; \hspace{1cm} $a := y_{acq}$

$y_{rel} := 1$;

if $a \neq 0$ then $b := x_{na}$

$$\{a = 0 \lor b = 5\}$$
Release-acquire synchronization: message passing

Let \( Q(v) \triangleq (v = 0 \lor x \mapsto 5) \).

\[
\begin{align*}
\{ x \mapsto 0 \} & \ast \{ y \mapsto 0 \} \\
\{ x \mapsto 0 \ast W_Q(y) \ast R_Q(y) \}
\end{align*}
\]

\( x_{na} := 5; \quad a := y_{acq} \)

\( y_{rel} := 1; \quad \text{if } a \neq 0 \text{ then } b := x_{na} \)

\( \{ a = 0 \lor b = 5 \} \)
Release-acquire synchronization: message passing

Let \( Q(v) \overset{△}{=} (v = 0 \lor x \mapsto 5) \).

\[
\begin{align*}
\{ x \mapsto 0 \ast y \mapsto 0 \} \\
\{ x \mapsto 0 \ast W_Q(y) \ast R_Q(y) \} \\
\{ x \mapsto 0 \ast W_Q(y) \} & \quad \{ R_Q(y) \} \\
x_{na} := 5; & \quad a := y_{acq}
\end{align*}
\]

\[
y_{rel} := 1; \quad \text{if } a \neq 0 \text{ then } b := x_{na}
\]

\[
\{ a = 0 \lor b = 5 \}
\]
Release-acquire synchronization: message passing

Let \( Q(v) \triangleq (v = 0 \lor x \mapsto 5). \)

\[
\begin{align*}
\{ x \mapsto 0 \ast y \mapsto 0 \} & \quad \{ x \mapsto 0 \ast W_Q(y) \ast R_Q(y) \} \\
\{ x \mapsto 0 \ast W_Q(y) \} & \quad \{ R_Q(y) \} \\
x_{na} := 5; & \quad a := y_{acq} \\
\{ x \mapsto 5 \ast W_Q(y) \} & \quad \text{if } a \neq 0 \text{ then } b := x_{na} \\
y_{rel} := 1; & \\
\{ a = 0 \lor b = 5 \} &
\end{align*}
\]
Let \( Q(\nu) \triangleq (\nu = 0 \lor x \mapsto 5). \)

\[
\begin{align*}
\{ x \mapsto 0 \} & \quad \{ y \mapsto 0 \} \\
\{ x \mapsto 0 \} & \quad \{ W_Q(y) \} \quad \{ \text{R}_Q(y) \} \\
x_{\text{na}} & := 5; \\
\{ x \mapsto 5 \} & \quad \{ W_Q(y) \} \\
y_{\text{rel}} & := 1; \\
\{ W_Q(y) \} & \quad \{ \text{R}_Q(y) \} \\
\text{if } a \neq 0 \text{ then } b & := x_{\text{na}} \\
\{ a = 0 \lor b = 5 \}
\end{align*}
\]
Release-acquire synchronization: message passing

Let \( Q(v) \triangleq (v = 0 \lor x \mapsto 5). \)

\[
\begin{align*}
\{ x \mapsto 0 \} \times \{ y \mapsto 0 \} \\
\{ x \mapsto 0 \} \times \{ W_Q(y) \} \times \{ R_Q(y) \} \\
\{ x \mapsto 0 \} \times \{ W_Q(y) \} \\
\{ x \mapsto 5 \} \times \{ W_Q(y) \} \\
\{ W_Q(y) \} \\
\{ \top \} \\
\{ a \mapsto y_{acq} \} \\
x_{na} := 5; \\
y_{rel} := 1; \\
\{ W_Q(y) \} \\
\{ \top \} \\
\{ a = 0 \lor b = 5 \}
\end{align*}
\]

\[
\begin{align*}
x_{na} := 5; \\
y_{rel} := 1; \\
\{ W_Q(y) \} \\
\{ \top \} \\
\{ a = 0 \lor b = 5 \}
\end{align*}
\]

\[
\begin{align*}
a := y_{acq} \\
\{ a \neq 0 \Rightarrow b := x_{na} \}
\end{align*}
\]
Release-acquire synchronization: message passing

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

\[
\begin{align*}
\{ x \mapsto 0 \ast y \mapsto 0 \} \\
\{ x \mapsto 0 \ast W_Q(y) \ast R_Q(y) \} \\
\{ x \mapsto 0 \ast W_Q(y) \} & \quad \{ R_Q(y) \} \\
\{ y_{\text{rel}} := 1; \} & \quad \{ (a = 0 \lor x \mapsto 5) \ast R_Q[a:=\text{emp}](y) \} \\
\{ W_Q(y) \} & \quad \{ a := y_{\text{acq}} \} \\
\{ \top \} & \quad \{ a \neq 0 \text{ then } b := x_{\text{na}} \}
\end{align*}
\]
Release-acquire synchronization: message passing

Let \( Q(v) \overset{\triangle}{=} (v = 0 \lor x \mapsto 5). \)

\[
\begin{align*}
\{x \mapsto 0 \ast y \mapsto 0\} & \quad \{x \mapsto 0 \ast W_Q(y) \ast R_Q(y)\} \\
\{x \mapsto 0 \ast W_Q(y)\} & \quad \{R_Q(y)\} \\
x_{na} := 5; & \quad a := y_{acq} \\
\{x \mapsto 5 \ast W_Q(y)\} & \quad (a = 0 \lor x \mapsto 5) \ast R_Q[a:=\text{emp}](y) \\
y_{rel} := 1; & \quad a = 0 \lor x \mapsto 5 \\
\{W_Q(y)\} & \quad \text{if } a \neq 0 \text{ then } b := x_{na} \\
\{\top\} & \quad \{a = 0 \lor b = 5\}
\end{align*}
\]
Release-acquire synchronization: message passing

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

$$
\begin{align*}
\{ x \mapsto 0 \} & \quad \{ y \mapsto 0 \} \\
\{ x \mapsto 0 \} & \quad \{ x \mapsto 0 \} \\
\{ x \mapsto 0 \} & \quad \{ R_Q(y) \} \\
\{ x \mapsto 5 \} & \quad \{ (a = 0 \lor x \mapsto 5) \} \\
\{ y_{\text{rel}} := 1 \} & \quad \{ a = 0 \lor x \mapsto 5 \} \\
\{ W_Q(y) \} & \quad \{ a = 0 \lor (x \mapsto 5 \land b = 5) \} \\
\{ x_{\text{na}} := 5 \} & \quad \{ a = 0 \lor b = 5 \} \\
\{ \top \} & \quad \{ a = 0 \lor b = 5 \}
\end{align*}
$$

Ownership transfer works!
Release-acquire synchronization: message passing

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

\[
\begin{align*}
\{x \mapsto 0 \land y \mapsto 0\} \\
\{x \mapsto 0 \land W_Q(y) \land R_Q(y)\}
\end{align*}
\]

\[
\begin{align*}
\{x \mapsto 0 \land W_Q(y)\} & \quad \{R_Q(y)\} \\
x_{na} := 5; & \quad a := y_{acq} \\
\{x \mapsto 5 \land W_Q(y)\} & \quad \{(a = 0 \lor x \mapsto 5) \land R_{Q[a:=emp]}(y)\} \\
y_{rel} := 1; & \quad \{a = 0 \lor x \mapsto 5\} \\
\{W_Q(y)\} & \quad \text{if } a \neq 0 \text{ then } b := x_{na} \\
\{\top\} & \quad \{a = 0 \lor (x \mapsto 5 \land b = 5)\} \\
\{a = 0 \lor b = 5\} & \quad \{a = 0 \lor b = 5\}
\end{align*}
\]

Ownership transfer works!
Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

\[
\{R_Q(x)\} \ a := x_{rlx} \ \{R_Q(x) \land (Q(a) \neq false)\}
\]

- Relaxed writes:

\[
\frac{Q(v) = \text{emp}}{\{W_Q(x)\} \ x_{rlx} := v \ \{W_Q(x)\}}
\]

Unsound because of dependency cycles!
Relaxed accesses

Basically, disallow ownership transfer.

- Relaxed reads:

\[
\{R_Q(x)\} \ a \ := \ x_{rlx} \ \{R_Q(x) \land (Q(a) \neq false)\}
\]

- Relaxed writes:

\[
Q(v) = emp \\
\{W_Q(x)\} \ x_{rlx} \ := \ v \ \{W_Q(x)\}
\]

Unsound because of dependency cycles!
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 $\ast$-ACI)

No (global) notions of state and time
- Define a *logical* local notion of state
- Annotate $\text{hb}$ edges with logical state

Declarative semantics
- Induct over max $\text{hb}$-path distance from top
Local annotation validity

For node $n$ roughly, \[ \sum_{e \in \text{in}(n)} hmap(e) + \text{effect}(n) \approx \sum_{e \in \text{out}(n)} hmap(e) \]

Release writes:

\[ W_{\text{rel}} \xrightarrow{h_F} h_F \oplus h_Q \]

\[ \downarrow \]

\[ h_F \oplus h_Q \]

\[ h_F \downarrow \]

\[ \text{R}_{\text{acq}} x, v \]

\[ \text{R}_{\text{acq}} x, v \]

\[ \text{h}_{1} \oplus \text{h}_{2} \oplus \text{h}_{3} = \text{h}_{Q} \]

\[ h_{F} \models \text{W}_{Q}(x) \ast \top \]

\[ h_{Q} \models \text{Q}(v) \]

Acquire reads:

\[ \text{R}_{\text{acq}} x, v \]

\[ \downarrow \]

\[ h_{F} \oplus h_{Q} \]

\[ h_{Q} \downarrow \]

\[ h_{F} \]

\[ h_{F} \models \text{R}_{Q}(x) \ast \top \]

\[ h_{Q} \models \text{Q}(v) \]
Independent heap compatibility

Definition (Pairwise independence)

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

Lemma (Independent heap compatibility)

If $hmap$ is a locally valid annotation of execution graph $G$ and $\mathcal{T} \subseteq G.\text{hb}$ 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 \( \{\text{true}\} \ c \ \{\text{true}\} \). Then, every consistent execution graph \( G \in \llbracket c \rrbracket \) has a valid annotation.

**Theorem (Race-Freedom)**

If \( \{\text{true}\} \ c \ \{\text{true}\}, \ \forall G \in \llbracket c \rrbracket, \ G \text{ is race-free.} \)
Fenced separation logic
Incorrect message passing

Initially $x = y = 0$.

\[ x_{\text{na}} := 5; \]
\[ y_{\text{rlx}} := 1 \]

\[
\begin{align*}
\text{repeat} \\
\quad a := y_{\text{rlx}} \\
\text{until } a \neq 0; \\
\quad b := x_{\text{na}}
\end{align*}
\]
Message passing with C11 memory fences

Initially $x = y = 0$.

Initially $x = y = 0$.

repeat
\[ a := y_{rlx} \]
until $a \neq 0$
\[ b := x_{na} \]

\[ x_{na} := 5; \]
\[ \text{fence}(\text{rel}); \]
\[ y_{rlx} := 1 \]

\[ W_{na} x, 5 \]
\[ F_{rel} \]
\[ R_{rlx} y, 1 \]
\[ W_{rlx} y, 1 \]
\[ F_{acq} \]
\[ R_{na} x, 5 \]
Fenced separation logic

Introduce two ‘modalities’ in the logic:

- △P : state ready to be transferred away.
- ▽P : state that will be acquired after a fence(acq).

Proof rules:

\[
\begin{align*}
\{P\} \quad \text{fence}(\text{rel}) & \quad \{△P\} \\
\{W_Q(x) \ast △Q(\nu)\} \quad x_{rlx} := \nu & \quad \{W_Q(x)\} \\
\{R_Q(x)\} \quad t := x_{rlx} & \quad \{R_{Q[t:=\text{emp}]}(x) \ast ▽Q(t)\} \\
\{▽P\} \quad \text{fence}(\text{acq}) & \quad \{P\}
\end{align*}
\]
Message passing with C11 memory fences

Let \( Q(v) \equiv v = 0 \lor x \mapsto 5. \)

\[
\begin{align*}
\{ x \mapsto 0 \} & \quad \{ y \mapsto 0 \} \\
\{ x \mapsto 0 \} & \quad \{ R_Q(y) \} \\
\{ x \mapsto 0 \} & \quad \{ W_Q(y) \} \\
\{ x \mapsto 0 \} & \quad \{ W_Q(y) \} \\
1 & \quad \{ a = 0 \lor (x \mapsto 5 \land b = 5) \}
\end{align*}
\]

\[
\begin{align*}
x_{na} & := 5; \\
\{ x \mapsto 5 \} & \quad \{ R_Q(y) \} \\
fence(\text{rel}); \\
\{ \triangle(x \mapsto 5) \} & \quad \{ \triangle(x \mapsto 5) \} \\
y_{rlx} & := 1 \\
\{ W_Q(y) \} & \quad \{ W_Q(y) \} \\
\{ a \neq 0 \} & \quad \{ a \neq 0 \} \\
\text{if } a \neq 0 & \quad \text{if } a \neq 0 \\
\{ x \mapsto 5 \} & \quad \{ x \mapsto 5 \} \\
fence(\text{acq}); \\
\{ x \mapsto 5 \} & \quad \{ x \mapsto 5 \} \\
b & := x_{na} \\
\{ x \mapsto 5 \} & \quad \{ x \mapsto 5 \} \\
\{ a = 0 \lor (x \mapsto 5 \land b = 5) \} & \quad \{ a = 0 \lor (x \mapsto 5 \land b = 5) \}
\end{align*}
\]
GPS
Three key features:

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

Example (Racy message passing)

Initially, $x = y = 0$.

\[
\begin{align*}
x_{rlx} & := 1; \quad \parallel \quad x_{rlx} := 1; \quad \parallel \quad a := y_{acq}; \\
y_{rel} & := 1 \quad \parallel \quad y_{rel} := 1 \quad \parallel \quad b := x_{rlx}
\end{align*}
\]

Cannot get $a = 1 \land b = 0$. 
Racy message passing in GPS

Protocol for $x$:  

| A: $x = 0$ | B: $x = 1$ |

Protocol for $y$:  

| C: $y = 0$ | D: $y = 1 \land x.st \geq B$ |

Acquire reads gain knowledge, not ownership.

\[
\begin{align*}
\{x.st \geq A \land y.st \geq C\} & \quad \{x.st \geq A \land y.st \geq C\} \\
x_{rlx} := 1; & \quad a := y_{acq}; \\
\{x.st \geq B \land y.st \geq C\} & \quad \{a = 0 \land x.st \geq A\} \\
y_{rel} := 1 & \quad \lor a = 1 \land x.st \geq B \\
\{x.st \geq B \land y.st \geq D\} & \quad b := x_{rlx}; \\
\{a = 0 \lor (a = 1 \land b = 1)\} & \quad \{a = 0 \lor (a = 1 \land b = 1)\}
\end{align*}
\]
GPS ghosts and escrows

To gain ownership, we use ghost state & escrows.

\[
P \ast P \Rightarrow false \\
Q \Rightarrow \text{Esc}(P, Q) \\
\text{Esc}(P, Q) \ast P \Rightarrow Q
\]

Example (Message passing using escrows)

Invariant for \( x \): \( x = 0 \lor \text{Esc}(K, \&a \mapsto 7) \).

\[
\begin{align*}
\{&a \mapsto 0\} \\
& a = 7; \\
\{&a \mapsto 7\} \\
\{\text{Esc}(K, \&a \mapsto 7)\} \\
x.\text{store}(1, \text{rel});
\end{align*}
\]

\[
\begin{align*}
\{K\} \\
\text{if} \ (x.\text{load}(\text{acq}) \neq 0) \\
\{K \ast \text{Esc}(K, \&a \mapsto 7)\} \\
\{&a \mapsto 7\} \\
\text{print}(a);
\end{align*}
\]
Challenge #1. Reasoning about SC atomics

SC fences

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

\[ J \ast P \ast P' \Rightarrow J \ast Q \ast Q' \]

\[ J \vdash \{ P \ast \triangledown P' \} \ fence(sc) \{ Q \ast \triangle Q' \} \]

SC accesses

- Program logics for SC $\sim$ 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 $\text{po} \cup \text{rf}$ acyclicity, which disallows the weak behaviour of LB.
- Even the logic is too strong.

**Load buffering (LB)**

Initially, $x = y = 0$

- $a := y$; \hspace{1em} $b := x$; \hspace{1em} $x := 1$; \hspace{1em} $y := 1$

**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
