Quarantining Weakness

Compositional Reasoning Under Relaxed Memory Models

Radha Jagadeesan* Gustavo Petri† Corin Pitcher* James Riely*

*DePaul University †Purdue University

ESOP 2013

Compositional Reasoning, Relaxed Memory

What is the interface of a concurrent object?

implicit causality via absence of interleavings . . .

What changes to account for weak memory?

. . . explicit causality via happens-before

Compositional Reasoning, Relaxed Memory

What is the interface of a concurrent object?

implicit causality via absence of interleavings ...

What changes to account for weak memory?

... explicit causality via happens-before

Spin lock (Library)

- Implementation

```
var v=1;                                        (Initially locked)
fun rel() { v=0; }                              (Strong memory)
fun acq() { do skip until v.cas(0,1); }
```

- Trace of release

$\underbrace{\langle ?\text{call rel}\rangle}_{\text{(Input) (Take control)}}\ \underbrace{\langle \text{wr v } 0\rangle}_{\text{(Write)}}\ \underbrace{\langle !\text{ ret rel}\rangle}_{\text{(Output) (Give control)}}$ (Lock's viewpoint)

- Trace of acquire

$\langle ?\text{call acq}\rangle\ \underbrace{\langle \text{rd v } 1\rangle\langle \text{rd v } 1\rangle \cdots \langle \text{rd v } 1\rangle}_{\text{(Unsuccessful cas treated as read)}}\ \langle \text{cas v } 0\ 1\rangle\langle !\text{ ret acq}\rangle$

Spin lock (Library)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Initially locked)

(Strong memory)

- Trace of release

$\langle ?\text{call rel}\rangle$ $\langle \text{wr v 0}\rangle$ $\langle !\text{ ret rel}\rangle$

(Lock's viewpoint)

$\langle ?\text{call rel}\rangle$: (Input) (Take control)

$\langle \text{wr v 0}\rangle$: (Write)

$\langle !\text{ ret rel}\rangle$: (Output) (Give control)

- Trace of acquire

$\langle ?\text{call acq}\rangle$ $\langle \text{rd v 1}\rangle\langle \text{rd v 1}\rangle \cdots \langle \text{rd v 1}\rangle$ $\langle \text{cas v 0 1}\rangle\langle !\text{ ret acq}\rangle$

$\langle \text{rd v 1}\rangle\langle \text{rd v 1}\rangle \cdots \langle \text{rd v 1}\rangle$: (Unsuccessful cas treated as read)

Spin lock (Library)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Initially locked)

(Strong memory)

- Trace of release

$\langle ?\text{call rel}\rangle$ (Input) (Take control) $\langle \text{wr v } 0\rangle$ (Write) $\langle !\text{ret rel}\rangle$ (Output) (Give control)

(Lock's viewpoint)

- Trace of acquire

$\langle ?\text{call acq}\rangle \underbrace{\langle \text{rd v } 1\rangle\langle \text{rd v } 1\rangle \cdots \langle \text{rd v } 1\rangle}_{\text{(Unsuccessful cas treated as read)}} \langle \text{cas v } 0\ 1\rangle\langle !\text{ret acq}\rangle$

Spin lock (Two threads)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Possible interleaving (Color = thread)

⟨?call acq⟩⟨rd v 1⟩⟨?call rel⟩⟨wr v 0⟩⟨cas v 0 1⟩⟨!ret acq⟩⟨!ret rel⟩

- Impossible interleaving

⟨?call acq⟩⟨rd v 1⟩⟨cas v 0 1⟩⟨!ret acq⟩⟨?call rel⟩⟨wr v 0⟩⟨!ret rel⟩

- Looking only at I/O actions:

⟨?call rel⟩ must precede ⟨!ret acq⟩

Spin lock (Two threads)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Possible interleaving (Color = thread)

  $\langle$?call acq$\rangle\langle$rd v 1$\rangle\langle$?call rel$\rangle\langle$wr v 0$\rangle\langle$cas v 0 1$\rangle\langle$!ret acq$\rangle\langle$!ret rel$\rangle$

- Impossible interleaving

  $\langle$?call acq$\rangle\langle$rd v 1$\rangle\langle$cas v 0 1$\rangle\langle$!ret acq$\rangle\langle$?call rel$\rangle\langle$wr v 0$\rangle\langle$!ret rel$\rangle$

- Looking only at I/O actions:

  $\langle$?call rel$\rangle$ must precede $\langle$!ret acq$\rangle$

Spin lock (Two threads)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Possible interleaving (Color = thread)

$\langle$?call acq$\rangle\langle$rd v 1$\rangle\langle$?call rel$\rangle\langle$wr v 0$\rangle\langle$cas v 0 1$\rangle\langle$!ret acq$\rangle\langle$!ret rel$\rangle$

- Impossible interleaving

$\langle$?call acq$\rangle\langle$rd v 1$\rangle\langle$cas v 0 1$\rangle\langle$!ret acq$\rangle\langle$?call rel$\rangle\langle$wr v 0$\rangle\langle$!ret rel$\rangle$

- Looking only at I/O actions:

$\langle$?call rel$\rangle$ must precede $\langle$!ret acq$\rangle$

Spin lock (Two threads)

- Implementation

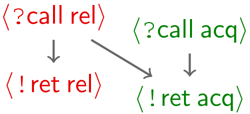
```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Abbreviate

$$\left\{\begin{array}{l}\langle ?\text{call rel}\rangle\langle !\text{ret rel}\rangle\langle ?\text{call acq}\rangle\langle !\text{ret acq}\rangle\\ \langle ?\text{call rel}\rangle\langle ?\text{call acq}\rangle\langle !\text{ret rel}\rangle\langle !\text{ret acq}\rangle\\ \langle ?\text{call rel}\rangle\langle ?\text{call acq}\rangle\langle !\text{ret acq}\rangle\langle !\text{ret rel}\rangle\\ \langle ?\text{call acq}\rangle\langle ?\text{call rel}\rangle\langle !\text{ret rel}\rangle\langle !\text{ret acq}\rangle\\ \langle ?\text{call acq}\rangle\langle ?\text{call rel}\rangle\langle !\text{ret acq}\rangle\langle !\text{ret rel}\rangle\end{array}\right\}$$

~~$\langle ?\text{call acq}\rangle\langle !\text{ret acq}\rangle\langle ?\text{call rel}\rangle\langle !\text{ret rel}\rangle$~~

as



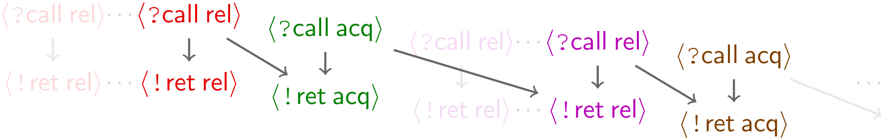
$\longrightarrow$ constrains interleavings

Spin lock (Two threads)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Interface: set of traces obeying



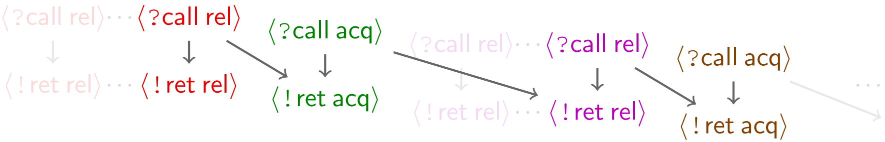
- ⟶ constrains interleavings
  ⟶ imposed by Lock
- What about client?

Spin lock (Two threads)

- Implementation

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

- Interface: set of traces obeying



- ⟶ constrains interleavings
  ⟶ imposed by Lock
- What about client?

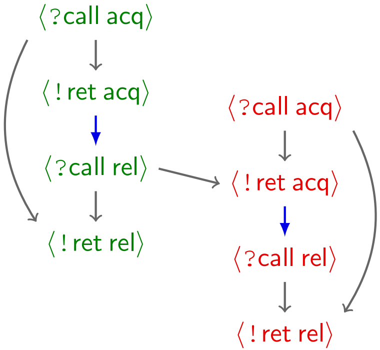
Spin lock (Client constraints)

- Implementation

```
var v=0;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Initially unlocked)

- Example of client order
  Multiple calls from single thread



Two kinds of constraints

? ⟶ !	Imposed by library (In → out)
! ⟶ ?	Imposed by client (Out → in)

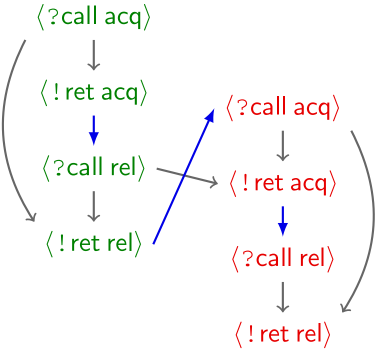
Spin lock (Client constraints)

- Implementation

```
var v=0;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Initially unlocked)

- Example of client order
  Multiple calls from single thread



Arrows constrain interleavings

More arrows = smaller set

Fully constrained = singleton, sequential

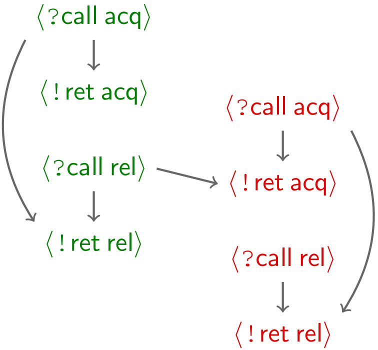
Spin lock (Client constraints)

- Implementation

```
var v=0;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Initially unlocked)

- Example of client order
  Multiple calls from single thread



Arrows constrain interleavings

More arrows = smaller set

No constraints = all allowed interleavings

Two locks (for one place buffer)

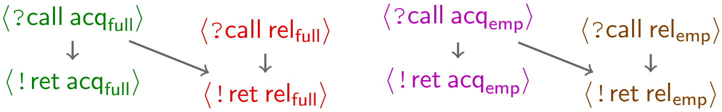
(Initially locked)

var v_{full}=1;
fun rel_{full} () { v_{full}=0; }
fun acq_{full} () { ⋯ v_{full}.cas(0, 1); }

(Initially unlocked)

var v_{emp}=0;
fun rel_{emp} () { v_{emp}=0; }
fun acq_{emp} () { ⋯ v_{emp}.cas(0, 1); }

- Locks are independent



- Client may create dependency

Two locks (for one place buffer)

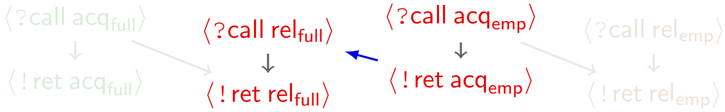
(Initially locked)

<u>var</u> v_{full}=1;
<u>fun</u> rel_{full} ( ) { v_{full}=0; }
<u>fun</u> acq_{full} ( ) { ⋯ v_{full} . cas (0, 1) ; }

(Initially unlocked)

<u>var</u> v_{emp}=0;
<u>fun</u> rel_{emp} ( ) { v_{emp}=0; }
<u>fun</u> acq_{emp} ( ) { ⋯ v_{emp} . cas (0, 1) ; }

- Locks are independent



- Client may create dependency

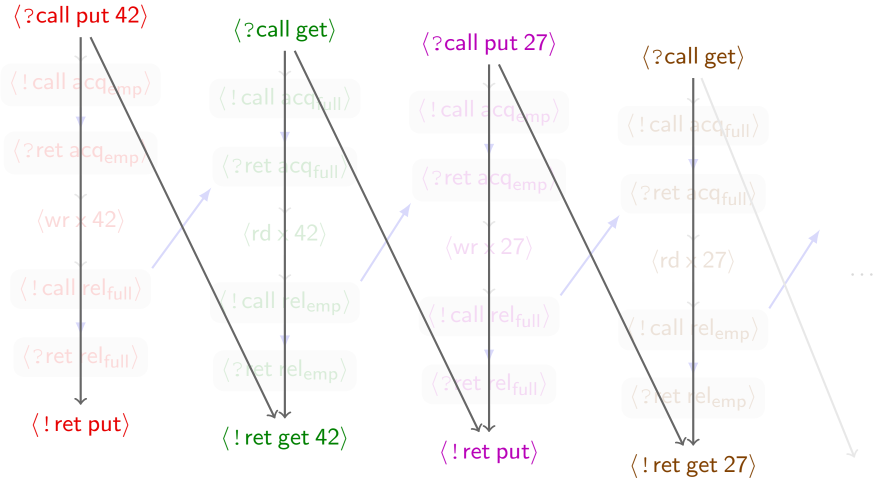
One place buffer (Initially empty)

```
var x=0;                                                              (r = register)
fun put (r)  { acq_emp (); x=r; rel_full (); }                        (emp unlocked)
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }          (full locked)
```

One place buffer (Initially empty)

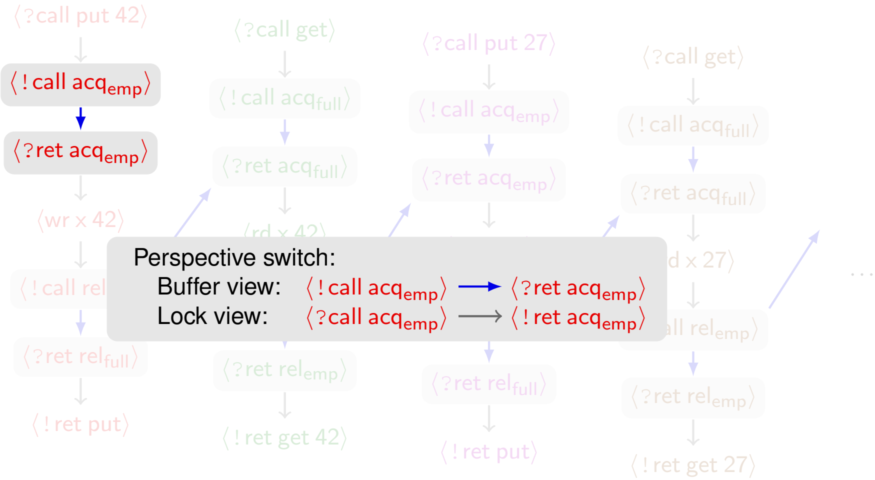
```
var x=0;
fun put (r) { acq_emp (); x=r; rel_full (); }
fun get ()  { acq_full (); let r=x; rel_emp (); return r; }
```

(r = register)
(emp unlocked)
(full locked)



One place buffer

```
var x=0;                                                       (r = register)
fun put (r)  { acq_emp (); x=r; rel_full (); }                 (emp unlocked)
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }   (full locked)
```



Perspective switch:

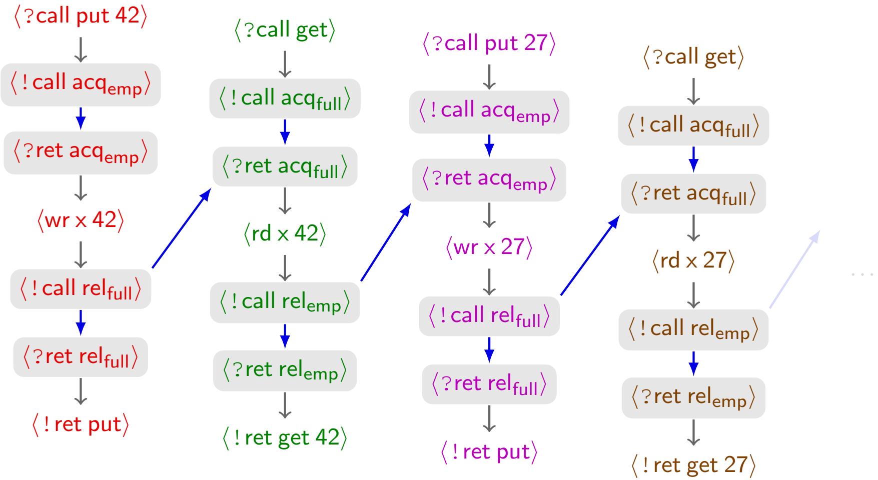
Buffer view: ⟨!call acq$_{emp}$⟩ → ⟨?ret acq$_{emp}$⟩

Lock view: ⟨?call acq$_{emp}$⟩ → ⟨!ret acq$_{emp}$⟩

One place buffer (Client interface)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```

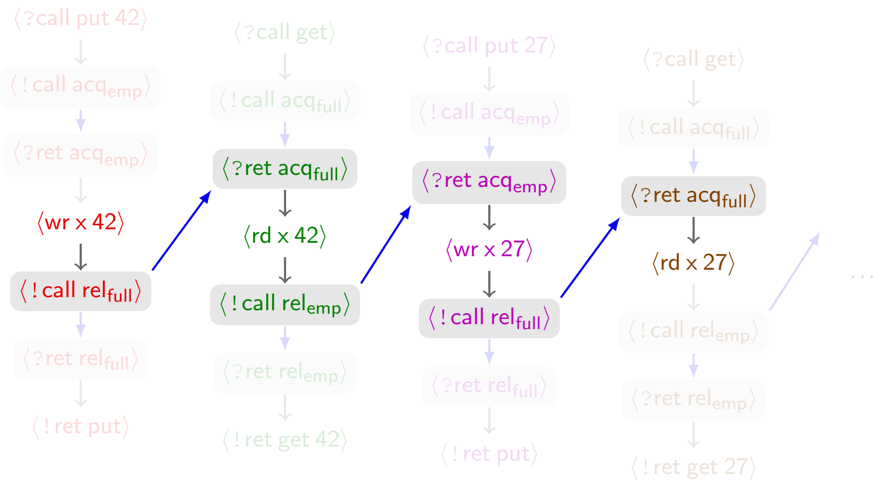
(r = register)
(emp unlocked)
(full locked)



One place buffer (Memory actions)

var x=0;
fun put (r) { acq_{emp} ( ) ; x=r; rel_{full} ( ) ; }
fun get ( ) { acq_{full} ( ) ; let r=x; rel_{emp} ( ) ; return r; }

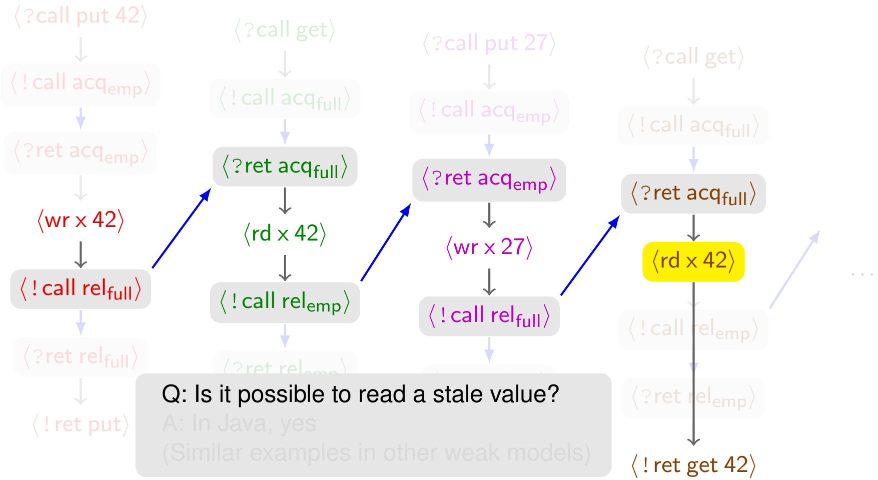
(r = register)
(emp unlocked)
(full locked)



One place buffer (Relaxed memory)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```

(r = register)
(emp unlocked)
(full locked)



Q: Is it possible to read a stale value?

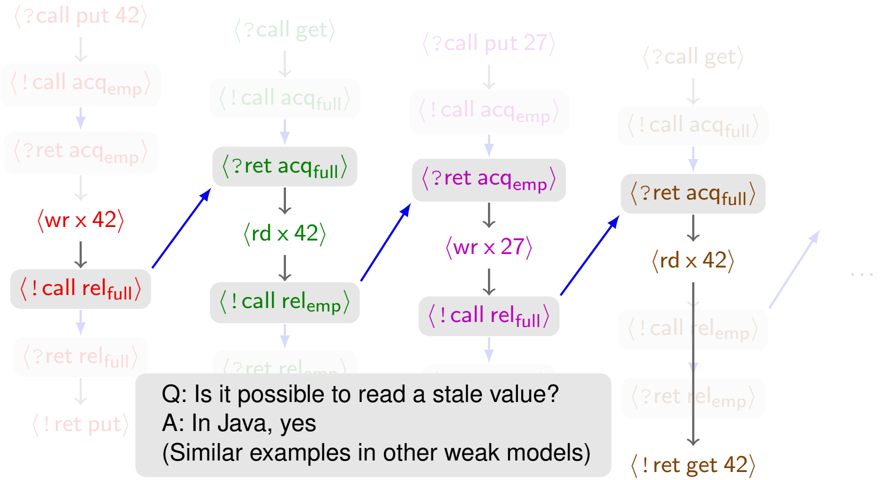
A: In Java, yes

(Similar examples in other weak models)

One place buffer (Relaxed memory)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```

(r = register)
(emp unlocked)
(full locked)

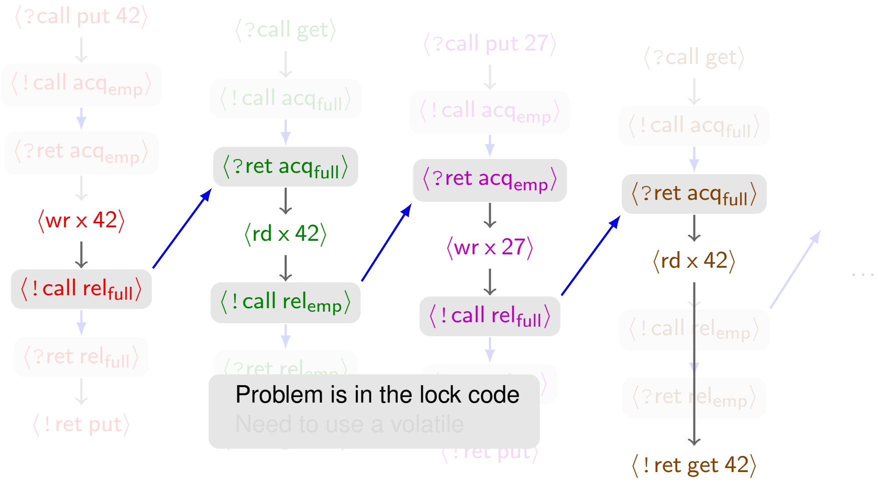


Q: Is it possible to read a stale value?
A: In Java, yes
(Similar examples in other weak models)

One place buffer (Relaxed memory)

```
var v=1;
fun rel() { v=0; }
fun acq() { do skip until v.cas(0,1); }
```

(Lock code)



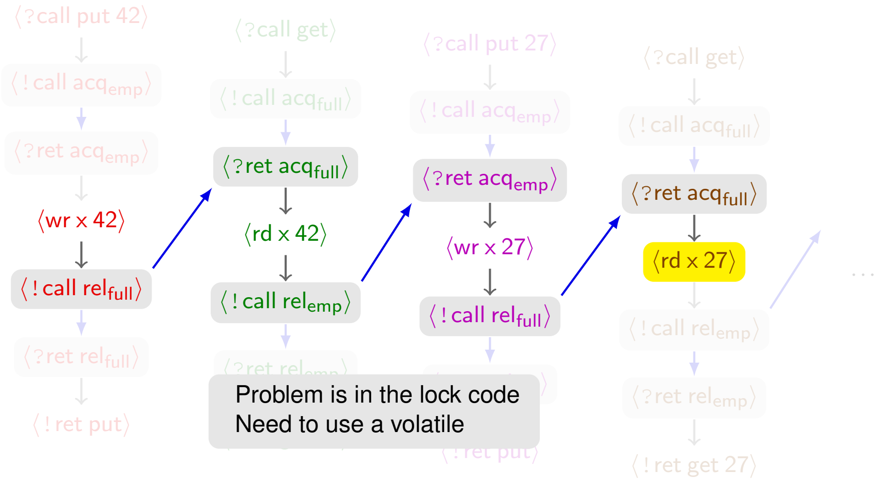
Problem is in the lock code

Need to use a volatile

One place buffer (Relaxed memory)

```
volatile v=1;
fun rel () { v=0; }
fun acq () { do skip until v.cas(0,1); }
```

(Lock code)



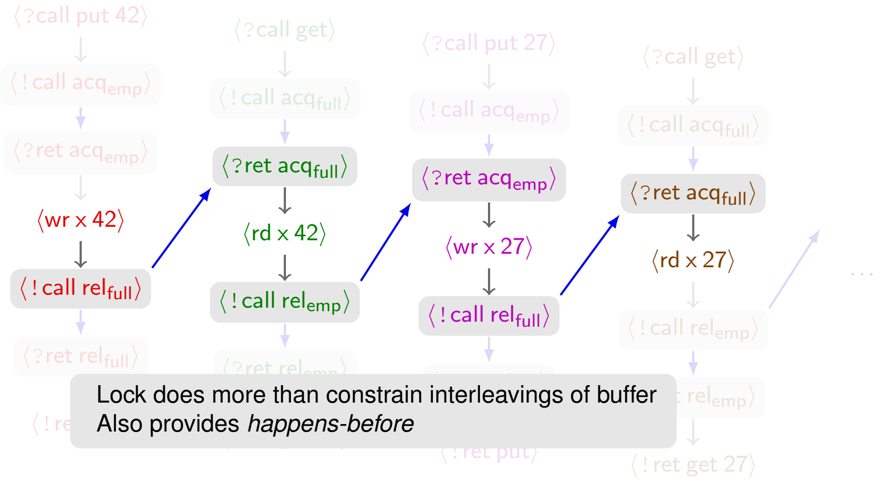
Problem is in the lock code
Need to use a volatile

One place buffer (Relaxed memory)

var x=0; (r = register)

fun put (r) { acq_{emp} ( ) ; x=r; rel_{full} ( ) ; } (emp unlocked)

fun get ( ) { acq_{full} ( ) ; let r=x; rel_{emp} ( ) ; return r; } (full locked)

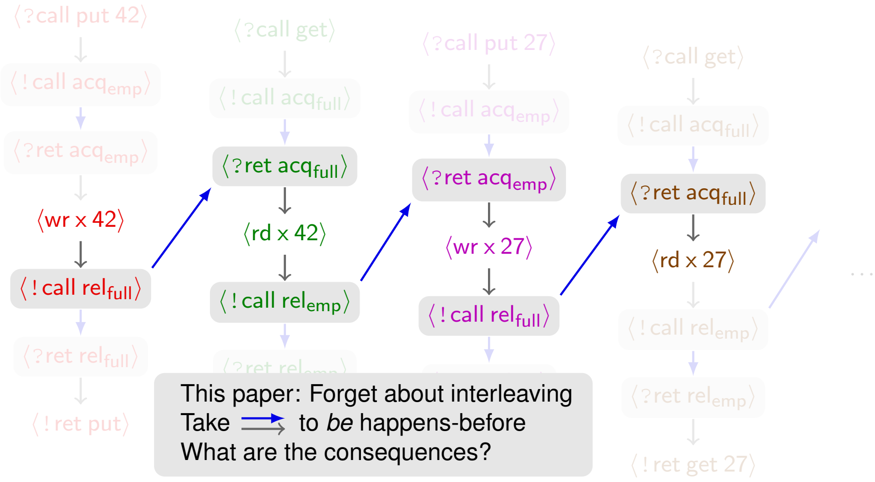


Lock does more than constrain interleavings of buffer
Also provides *happens-before*

One place buffer (Relaxed memory)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```

(r = register)
(emp unlocked)
(full locked)



This paper: Forget about interleaving
Take ⟶ to *be* happens-before
What are the consequences?

Plan

Traditional notions of correctness

Happens-before

Results

Compositionality

Notions of correctness

- Sequential consistency (SC) = methods appear atomic
  (Lamport, IEEE Trans. Comput. 1979)

$$(\forall \sigma \in Impl)\ (\exists \phi \in SequentialSpec)\ (\forall s \in Thread)\ \sigma|_s = \phi|_s$$

- Linearizability = Serializability + compositionality
  (Herlihy/Wing, POPL 1987, TOPLAS 1990)

  $\cdots$ and ϕ must respect order of nonoverlapping calls in σ

- Example ($Impl \sqsubseteq Spec$)

That is,

$$\{\langle ?\text{call } f\rangle\langle !\text{ret } f\rangle\langle ?\text{call } g\rangle\langle !\text{ret } g\rangle\} \quad \sqsubseteq \quad \{\langle ?\text{call } g\rangle\langle !\text{ret } g\rangle\langle ?\text{call } f\rangle\langle !\text{ret } f\rangle\}$$

Notions of correctness

- Serializability = methods appear atomic
  (Lamport, IEEE Trans. Comput. 1979)

$$(\forall \sigma \in \mathit{Impl})\ (\exists \phi \in \mathit{SequentialSpec})\ (\forall s \in \mathit{Thread})\ \sigma|_s = \phi|_s$$

- Linearizability = Serializability + compositionality
  (Herlihy/Wing, POPL 1987, TOPLAS 1990)

  $\cdots$ and ϕ must respect order of nonoverlapping calls in σ

- Example ($\mathit{Impl} \sqsubseteq \mathit{Spec}$)

That is,

$$\{\langle ?\mathsf{call}\ f\rangle\langle !\,\mathsf{ret}\ f\rangle\langle ?\mathsf{call}\ g\rangle\langle !\,\mathsf{ret}\ g\rangle\} \quad \sqsubseteq \quad \{\langle ?\mathsf{call}\ g\rangle\langle !\,\mathsf{ret}\ g\rangle\langle ?\mathsf{call}\ f\rangle\langle !\,\mathsf{ret}\ f\rangle\}$$

Notions of correctness

- Serializability = methods appear atomic
  (Lamport, IEEE Trans. Comput. 1979)

$$(\forall \sigma \in \mathit{Impl})\ (\exists \phi \in \mathit{SequentialSpec})\ (\forall s \in \mathit{Thread})\ \sigma|_s = \phi|_s$$

- Linearizability = Serializability + compositionality
  (Herlihy/Wing, POPL 1987, TOPLAS 1990)

$\cdots$ and ϕ must respect order of nonoverlapping calls in σ

- Example ($\mathit{Impl} \sqsubseteq \mathit{Spec}$)

That is,

$$\{\langle ?\mathsf{call}\ f\rangle\langle !\mathsf{ret}\ f\rangle\langle ?\mathsf{call}\ g\rangle\langle !\mathsf{ret}\ g\rangle\} \quad \sqsubseteq \quad \{\langle ?\mathsf{call}\ g\rangle\langle !\mathsf{ret}\ g\rangle\langle ?\mathsf{call}\ f\rangle\langle !\mathsf{ret}\ f\rangle\}$$

Notions of correctness

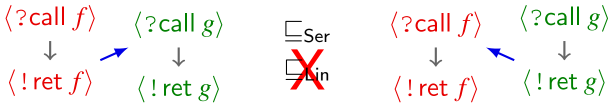
- Serializability = methods appear atomic
  (Lamport, IEEE Trans. Comput. 1979)

  $$(\forall \sigma \in \mathit{Impl})\ (\exists \phi \in \mathit{SequentialSpec})\ (\forall s \in \mathit{Thread})\ \sigma|_s = \phi|_s$$

- Linearizability = Serializability + compositionality
  (Herlihy/Wing, POPL 1987, TOPLAS 1990)

  $\cdots$ and ϕ must respect order of nonoverlapping calls in σ

- Example ($\mathit{Impl} \sqsubseteq \mathit{Spec}$)



That is,

$$\{\langle ?\mathsf{call}\ f\rangle\langle !\,\mathsf{ret}\ f\rangle\langle ?\mathsf{call}\ g\rangle\langle !\,\mathsf{ret}\ g\rangle\} \quad \sqsubseteq \quad \{\langle ?\mathsf{call}\ g\rangle\langle !\,\mathsf{ret}\ g\rangle\langle ?\mathsf{call}\ f\rangle\langle !\,\mathsf{ret}\ f\rangle\}$$

Philosophy

- Serializability and Linearizability
  (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009, TCS 2010)
  - Standard view

    "the illusion of atomicity"
  - Alternate view

    "conservative over-approximations of dependencies . . .
    that may arise in some client programs"

- Strong memory: views are interderivable

  global store = global clock

- Weak memory: views are distinct

  no global store = no global clock

temporal coincidence $\neq$ causality

Philosophy

- Serializability and Linearizability
  (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009, TCS 2010)
  - Standard view

    "the illusion of atomicity"

  - Alternate view

    "conservative over-approximations of dependencies . . .
    that may arise in some client programs"

- Strong memory: views are interderivable

  global store = global clock

- Weak memory: views are distinct

  no global store = no global clock

temporal coincidence $\neq$ causality

Happens-before

- Semantics as sets Σ, Φ of traces σ, ϕ with named actions
  - Four memory models: $\mathscr{W} \in \{\mathsf{strong}, \mathsf{tso}, \mathsf{pso}, \mathsf{jmm}\}$
  - Order recovered by relation $i <^{\sigma}_{\mathscr{W}} j$
  - Informally $(<^{\sigma}_{\mathscr{W}}) = (\Longrightarrow)$ (only one relation, color distinguishes polarity)
- In a specification:

$$\langle ?\mathsf{call}\ f \rangle \longrightarrow \langle !\,\mathsf{ret}\ g \rangle \text{ if } \langle ?\mathsf{call}\ f\ a \rangle \cdots \langle !\,\mathsf{ret}\ g\ \{a\} \rangle$$
$$\langle !\,\mathsf{ret}\ f \rangle \longrightarrow \langle ?\mathsf{call}\ g \rangle \text{ if } \langle !\,\mathsf{ret}\ f\ b \rangle \cdots \langle ?\mathsf{call}\ g\ \{b\} \rangle$$

- In opsem (thread s, actions a, b, volatile v):

$$\langle s\ a \rangle \longrightarrow \langle s\ b \rangle \text{ if } \langle s\ a \rangle \cdots \langle s\ b \rangle \quad \text{(thread order)}$$
$$\langle \mathsf{wr}\ v \rangle \longrightarrow \langle \mathsf{rd}\ v \rangle \text{ if } \langle \mathsf{wr}\ v \rangle \cdots \langle \mathsf{rd}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{wr}\ v \rangle \longrightarrow \langle \mathsf{cas}\ v \rangle \text{ if } \langle \mathsf{wr}\ v \rangle \cdots \langle \mathsf{cas}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v \rangle \longrightarrow \langle \mathsf{rd}\ v \rangle \text{ if } \langle \mathsf{cas}\ v \rangle \cdots \langle \mathsf{rd}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v \rangle \longrightarrow \langle \mathsf{cas}\ v \rangle \text{ if } \langle \mathsf{cas}\ v \rangle \cdots \langle \mathsf{cas}\ v \rangle \quad \text{(synchronization)}$$

- This defines $<^{\sigma}_{\mathsf{jmm}}$
  - $<^{\sigma}_{\mathsf{strong}}$ includes conflicts on all variables
  - $<^{\sigma}_{\mathsf{tso}}$ and $<^{\sigma}_{\mathsf{pso}}$ in between

Happens-before

- Semantics as sets Σ, Φ of traces σ, ϕ with named actions
  - Four memory models: $\mathscr{W} \in \{\mathsf{strong}, \mathsf{tso}, \mathsf{pso}, \mathsf{jmm}\}$
  - Order recovered by relation $i <^{\sigma}_{\mathscr{W}} j$
  - Informally $(<^{\sigma}_{\mathscr{W}}) = (\rightrightarrows)$ (only one relation, color distinguishes polarity)
- In a specification:

$$\langle ?\mathsf{call}\ f \rangle \longrightarrow \langle !\,\mathsf{ret}\ g \rangle \text{ if } \langle ?\mathsf{call}\ f\ a \rangle \cdots \langle !\,\mathsf{ret}\ g\ \{a\} \rangle$$
$$\langle !\,\mathsf{ret}\ f \rangle \longrightarrow \langle ?\mathsf{call}\ g \rangle \text{ if } \langle !\,\mathsf{ret}\ f\ b \rangle \cdots \langle ?\mathsf{call}\ g\ \{b\} \rangle$$

- In opsem (thread s, actions a, b, volatile v):

$$\langle s\ \ a \rangle \longrightarrow \langle s\ \ b \rangle \text{ if } \langle s\ \ a \rangle \cdots \langle s\ \ b \rangle \quad \text{(thread order)}$$
$$\langle \mathsf{wr}\ v \rangle \longrightarrow \langle \mathsf{rd}\ v \rangle \text{ if } \langle \mathsf{wr}\ v \rangle \cdots \langle \mathsf{rd}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{wr}\ v \rangle \longrightarrow \langle \mathsf{cas}\ v \rangle \text{ if } \langle \mathsf{wr}\ v \rangle \cdots \langle \mathsf{cas}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v \rangle \longrightarrow \langle \mathsf{rd}\ v \rangle \text{ if } \langle \mathsf{cas}\ v \rangle \cdots \langle \mathsf{rd}\ v \rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v \rangle \longrightarrow \langle \mathsf{cas}\ v \rangle \text{ if } \langle \mathsf{cas}\ v \rangle \cdots \langle \mathsf{cas}\ v \rangle \quad \text{(synchronization)}$$

- This defines $<^{\sigma}_{\mathsf{jmm}}$
  - $<^{\sigma}_{\mathsf{strong}}$ includes conflicts on all variables
  - $<^{\sigma}_{\mathsf{tso}}$ and $<^{\sigma}_{\mathsf{pso}}$ in between

Happens-before

- Semantics as sets Σ, Φ of traces σ, ϕ with named actions
  - Four memory models: $\mathscr{W} \in \{\mathsf{strong}, \mathsf{tso}, \mathsf{pso}, \mathsf{jmm}\}$
  - Order recovered by relation $i <^{\sigma}_{\mathscr{W}} j$
  - Informally $(<^{\sigma}_{\mathscr{W}}) = (\rightrightarrows)$ (only one relation, color distinguishes polarity)
- In a specification:

$$\langle ?\mathsf{call}\ f\rangle \longrightarrow \langle !\,\mathsf{ret}\ g\rangle \text{ if } \langle ?\mathsf{call}\ f\ a\rangle \cdots \langle !\,\mathsf{ret}\ g\ \{a\}\rangle$$
$$\langle !\,\mathsf{ret}\ f\rangle \longrightarrow \langle ?\mathsf{call}\ g\rangle \text{ if } \langle !\,\mathsf{ret}\ f\ b\rangle \cdots \langle ?\mathsf{call}\ g\ \{b\}\rangle$$

- In opsem (thread s, actions a, b, volatile v):

$$\langle s\ a\rangle \longrightarrow \langle s\ b\rangle \text{ if } \langle s\ a\rangle \cdots \langle s\ b\rangle \quad \text{(thread order)}$$
$$\langle \mathsf{wr}\ v\rangle \longrightarrow \langle \mathsf{rd}\ v\rangle \text{ if } \langle \mathsf{wr}\ v\rangle \cdots \langle \mathsf{rd}\ v\rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{wr}\ v\rangle \longrightarrow \langle \mathsf{cas}\ v\rangle \text{ if } \langle \mathsf{wr}\ v\rangle \cdots \langle \mathsf{cas}\ v\rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v\rangle \longrightarrow \langle \mathsf{rd}\ v\rangle \text{ if } \langle \mathsf{cas}\ v\rangle \cdots \langle \mathsf{rd}\ v\rangle \quad \text{(synchronization)}$$
$$\langle \mathsf{cas}\ v\rangle \longrightarrow \langle \mathsf{cas}\ v\rangle \text{ if } \langle \mathsf{cas}\ v\rangle \cdots \langle \mathsf{cas}\ v\rangle \quad \text{(synchronization)}$$

- This defines $<^{\sigma}_{\mathsf{jmm}}$
  - $<^{\sigma}_{\mathsf{strong}}$ includes conflicts on all variables
  - $<^{\sigma}_{\mathsf{tso}}$ and $<^{\sigma}_{\mathsf{pso}}$ in between

Happens-before

- Semantics as sets Σ, Φ of traces σ, ϕ with named actions
  - Four memory models: $\mathscr{W} \in \{\mathsf{strong}, \mathsf{tso}, \mathsf{pso}, \mathsf{jmm}\}$
  - Order recovered by relation $i <^{\sigma}_{\mathscr{W}} j$
  - Informally $(<^{\sigma}_{\mathscr{W}}) = (\rightrightarrows)$ (only one relation, color distinguishes polarity)
- In a specification:

$$\begin{array}{llll} \langle ?\mathsf{call}\ f\rangle & \longrightarrow & \langle !\,\mathsf{ret}\ g\rangle & \text{if } \langle ?\mathsf{call}\ f\ a\rangle \cdots \langle !\,\mathsf{ret}\ g\ \{a\}\rangle \\ \langle !\,\mathsf{ret}\ f\rangle & \longrightarrow & \langle ?\mathsf{call}\ g\rangle & \text{if } \langle !\,\mathsf{ret}\ f\ b\rangle \cdots \langle ?\mathsf{call}\ g\ \{b\}\rangle \end{array}$$

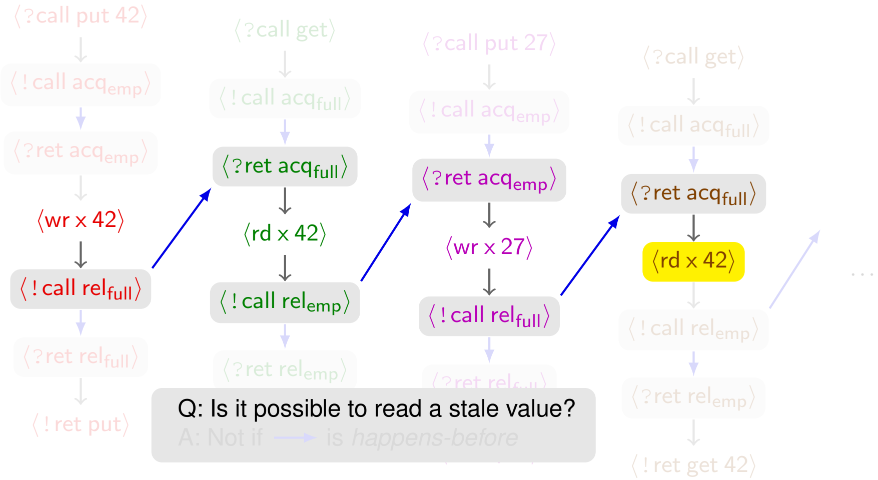
- In opsem (thread s, actions a, b, volatile v):

$$\begin{array}{lllll} \langle s\ \ a\rangle & \longrightarrow & \langle s\ \ b\rangle & \text{if } \langle s\ \ a\rangle \cdots \langle s\ \ b\rangle & \text{(thread order)} \\ \langle \mathsf{wr}\ v\rangle & \longrightarrow & \langle \mathsf{rd}\ v\rangle & \text{if } \langle \mathsf{wr}\ v\rangle \cdots \langle \mathsf{rd}\ v\rangle & \text{(synchronization)} \\ \langle \mathsf{wr}\ v\rangle & \longrightarrow & \langle \mathsf{cas}\ v\rangle & \text{if } \langle \mathsf{wr}\ v\rangle \cdots \langle \mathsf{cas}\ v\rangle & \text{(synchronization)} \\ \langle \mathsf{cas}\ v\rangle & \longrightarrow & \langle \mathsf{rd}\ v\rangle & \text{if } \langle \mathsf{cas}\ v\rangle \cdots \langle \mathsf{rd}\ v\rangle & \text{(synchronization)} \\ \langle \mathsf{cas}\ v\rangle & \longrightarrow & \langle \mathsf{cas}\ v\rangle & \text{if } \langle \mathsf{cas}\ v\rangle \cdots \langle \mathsf{cas}\ v\rangle & \text{(synchronization)} \end{array}$$

- This defines $<^{\sigma}_{\mathsf{jmm}}$
  - $<^{\sigma}_{\mathsf{strong}}$ includes conflicts on all variables
  - $<^{\sigma}_{\mathsf{tso}}$ and $<^{\sigma}_{\mathsf{pso}}$ in between

One place buffer (Revisited)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```

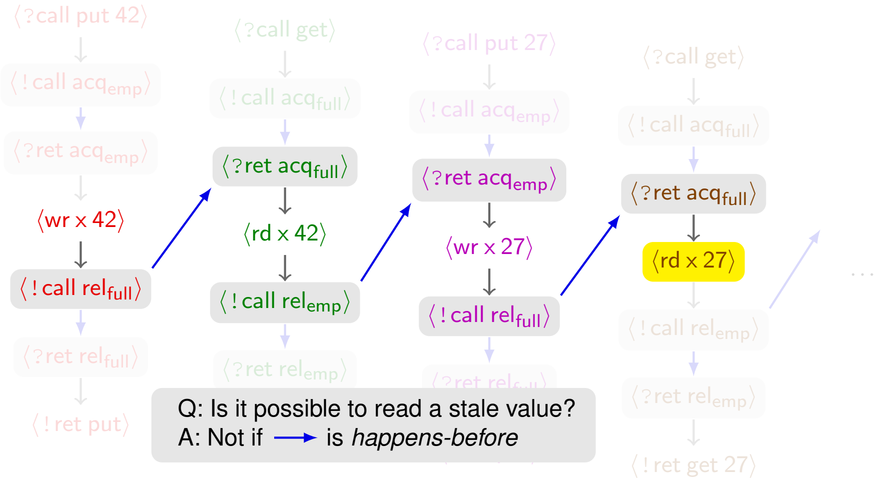


Q: Is it possible to read a stale value?

A: Not if ⟶ is *happens-before*

One place buffer (Revisited)

```
var x=0;
fun put (r)  { acq_emp (); x=r; rel_full (); }
fun get ()   { acq_full (); let r=x; rel_emp (); return r; }
```



Q: Is it possible to read a stale value?
A: Not if ⟶ is *happens-before*

Details

Define $\Sigma \sqsubseteq_{\mathsf{Lin}} \Phi$ as

$\forall \sigma \in \Sigma.$

$\exists \phi \in \Phi.$

$\exists \pi : [1 \ldots |\sigma|] \to [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathsf{thrd}} \pi(j)$ then $i < j$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $i <^{\sigma}_{\mathsf{thrd}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ and $i < j$ then $\pi(i) < \pi(j)$

σ has same I/O actions as ϕ

σ has same thread order as ϕ

nonoverlapping order of σ respected by ϕ

Details

Define $\Sigma \sqsubseteq_{\mathsf{Lin}} \Phi$ as

$\forall \sigma \in \Sigma.$
$\exists \phi \in \Phi.$
$\exists \pi : [1 \ldots |\sigma|] \to [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathsf{thrd}} \pi(j)$ then $i < j$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $i <^{\sigma}_{\mathsf{thrd}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ and $i < j$ then $\pi(i) < \pi(j)$

σ has same I/O actions as ϕ

σ has same thread order as ϕ

nonoverlapping order of σ respected by ϕ

Details

Define $\Sigma \sqsubseteq_{\mathsf{Lin}} \Phi$ as

$\forall \sigma \in \Sigma.$
$\exists \phi \in \Phi.$
$\exists \pi : [1 \ldots |\sigma|] \to [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathsf{thrd}} \pi(j)$ then $i < j$

if σ_i, σ_j are $\langle ? \rangle, \langle ! \rangle$ and $i <^{\sigma}_{\mathsf{thrd}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ and $i < j$ then $\pi(i) < \pi(j)$

σ has same I/O actions as ϕ

σ has same thread order as ϕ

nonoverlapping order of σ respected by ϕ

Details

Define $\Sigma \sqsubseteq_{\mathscr{W}} \Phi$ as

$\forall \sigma \in \Sigma.$

$\exists \phi \in \Phi.$

$\exists \pi : [1 \ldots |\sigma|] \to [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ then $i <^{\sigma}_{\mathscr{W}} j$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $i <^{\sigma}_{\mathscr{W}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ then $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ iff $i <^{\sigma}_{\mathscr{W}} j$

σ has same I/O actions as ϕ

σ has more $\longrightarrow$ than ϕ

extra order of σ does not contradict ϕ

σ has same $\longrightarrow$ as ϕ

Details

Define $\Sigma \sqsubseteq_{\mathscr{W}} \Phi$ as

$\forall \sigma \in \Sigma.$
$\exists \phi \in \Phi.$
$\exists \pi : [1 \ldots |\sigma|] \to [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ then $i <^{\sigma}_{\mathscr{W}} j$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $i <^{\sigma}_{\mathscr{W}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ then $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ iff $i <^{\sigma}_{\mathscr{W}} j$

σ has same I/O actions as ϕ

σ has more $\longrightarrow$ than ϕ

extra order of σ does not contradict ϕ

σ has same $\longrightarrow$ as ϕ

Details

Define $\Sigma \sqsubseteq_{\mathscr{W}} \Phi$ as
$\forall \sigma \in \Sigma.$
$\exists \phi \in \Phi.$
$\exists \pi : [1 \ldots |\sigma|] \rightarrow [1 \ldots |\phi|].$

if either $\sigma_i, \sigma_{\pi(i)}$ is $\langle ? \rangle, \langle ! \rangle$ then $\sigma_i = \phi_{\pi(i)}$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ then $i <^{\sigma}_{\mathscr{W}} j$

if $\sigma_i = \langle ? \rangle$, $\sigma_j = \langle ! \rangle$ and $i <^{\sigma}_{\mathscr{W}} j$ then $\pi(i) < \pi(j)$

if $\sigma_i = \langle ! \rangle$, $\sigma_j = \langle ? \rangle$ then $\pi(i) <^{\phi}_{\mathscr{W}} \pi(j)$ iff $i <^{\sigma}_{\mathscr{W}} j$

σ has same I/O actions as ϕ

σ has more $\longrightarrow$ than ϕ

extra order of σ does not contradict ϕ

σ has same $\longrightarrow$ as ϕ

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

  $\sqsubseteq$ = linearizability

  $\subseteq$ = observational refinement

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

  "client" P

  "library" impl Σ, spec Φ

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

  Operational composition

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Refinement theorems (client and library have disjoint variables)

- Theorem (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- Theorem (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

  Operational composition
  Operational spec

- Theorem (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$ — Arbitrary spec for library

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\text{strong}} \Phi$
  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\text{impl}}]\!] \sqsubseteq_{\text{tso}} [\![Q_{\text{spec}}]\!]$
  then $[\![P \parallel Q_{\text{impl}}]\!] \subseteq [\![P \parallel Q_{\text{spec}}]\!]$

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$
  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$
  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

  Arbitrary spec for library
  Explicit tensor

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

- **Theorem** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathscr{W}} \Phi_P$

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

if library satisfies spec
and client correct using spec
then composed system correct
$\mathscr{W} \in \{\mathsf{strong}, \mathsf{tso}, \mathsf{pso}, \mathsf{jmm}\}$

Refinement theorems (client and library have disjoint variables)

- **Theorem** (Filipović/O'Hearn/Rinetzky/Yang, ESOP 2009)

  if $\Sigma \sqsubseteq_{\mathsf{strong}} \Phi$

  then $[\![P]\!](\Sigma) \subseteq [\![P]\!](\Phi)$

- **Theorem** (Burckhardt/Gotsman/Musuvathi/Yang, ESOP 2012)

  if $[\![Q_{\mathsf{impl}}]\!] \sqsubseteq_{\mathsf{tso}} [\![Q_{\mathsf{spec}}]\!]$

  then $[\![P \parallel Q_{\mathsf{impl}}]\!] \subseteq [\![P \parallel Q_{\mathsf{spec}}]\!]$

- **Corollary** (This paper)

  if $[\![Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_Q$

  and $[\![P]\!] \otimes \Phi_Q \sqsubseteq_{\mathsf{strong}} \sqsubseteq_{\mathscr{W}} \Phi_P$

  and $[\![P]\!]$ is locally SC, ...

  then $[\![P \parallel Q]\!] \sqsubseteq_{\mathscr{W}} \Phi_P$

Well synchronized clients are not affected by races in library
(Details in proceedings)

Compositionality

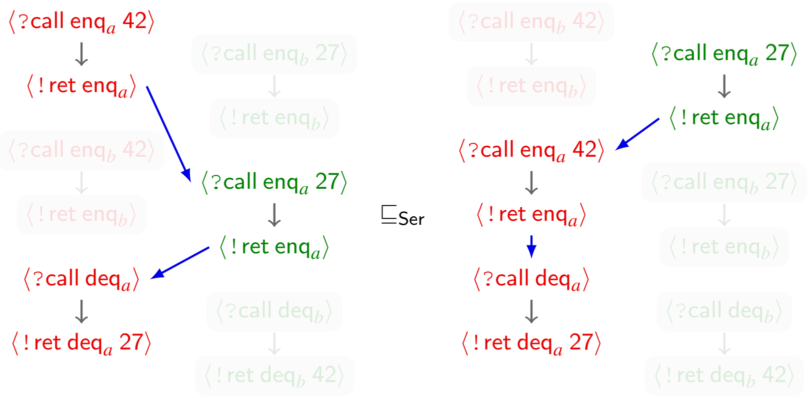
if $[\![P_i]\!] \sqsubseteq_{\mathscr{W}} \Sigma_i$ then $[\![P_1 \parallel P_2]\!] \sqsubseteq_{\mathscr{W}} \Sigma_1 \otimes \Sigma_2$

When does it hold?

(P_i have disjoint variables)
(Not a corollary of refinement)

Compositionality counterexample (Herlihy/Wing)

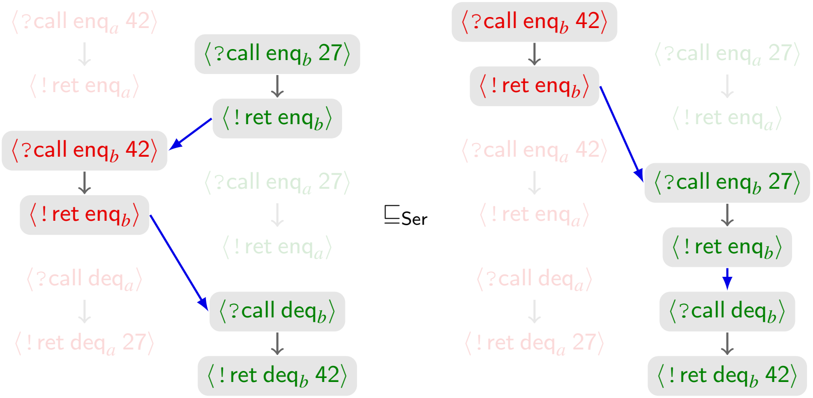
Serializable trace:



→ = “non-overlapping”: return before call.

Compositionality counterexample (Herlihy/Wing)

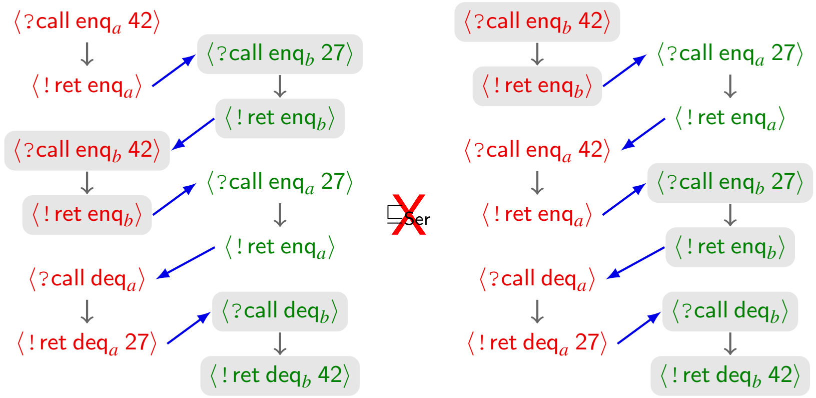
Serializable trace:



⟶ = “non-overlapping”: return before call.

Compositionality counterexample (Herlihy/Wing)

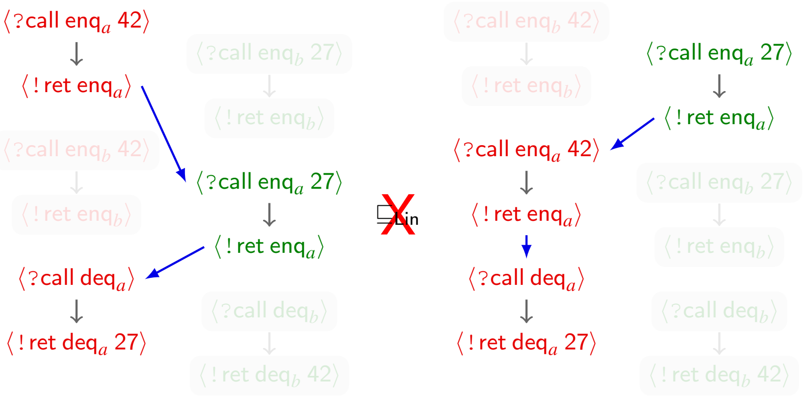
Non-Serializable trace:



→ = “non-overlapping”: return before call.

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:



→ = "non-overlapping": return before call.

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:

$\langle ?\text{call enq}_a\ 42\rangle \rightarrow \langle !\text{ret enq}_a\rangle$
$\langle ?\text{call enq}_b\ 42\rangle \rightarrow \langle !\text{ret enq}_b\rangle$
$\rightarrow \langle ?\text{call deq}_a\rangle \rightarrow \langle !\text{ret deq}_a\ 27\rangle$

$\langle ?\text{call enq}_b\ 27\rangle \rightarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call enq}_a\ 27\rangle \rightarrow \langle !\text{ret enq}_a\rangle$
$\langle ?\text{call deq}_b\rangle \rightarrow \langle !\text{ret deq}_b\ 42\rangle$

$\not\sqsubseteq_{\text{Lin}}$

$\langle ?\text{call enq}_b\ 42\rangle \rightarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call enq}_a\ 42\rangle \rightarrow \langle !\text{ret enq}_a\rangle \rightarrow \langle ?\text{call deq}_a\rangle \rightarrow \langle !\text{ret deq}_a\ 27\rangle$

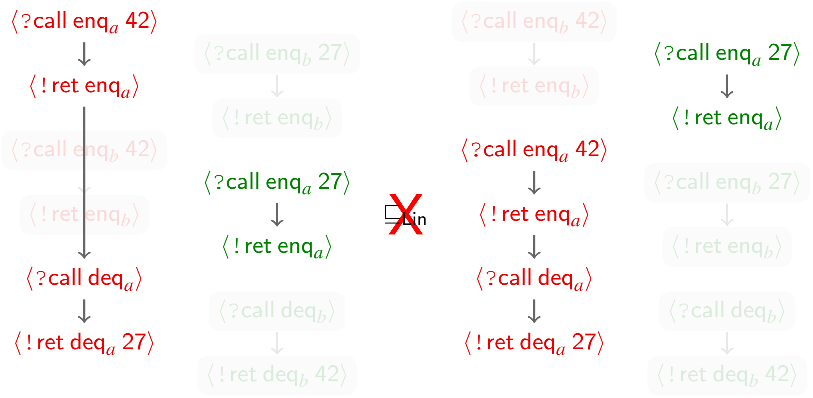
$\langle ?\text{call enq}_a\ 27\rangle \rightarrow \langle !\text{ret enq}_a\rangle$
$\langle ?\text{call enq}_b\ 27\rangle \rightarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call deq}_b\rangle \rightarrow \langle !\text{ret deq}_b\ 42\rangle$

Crucial point:

$$\{\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \not\sqsubseteq_{\text{Lin}} \{\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\}$$

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:

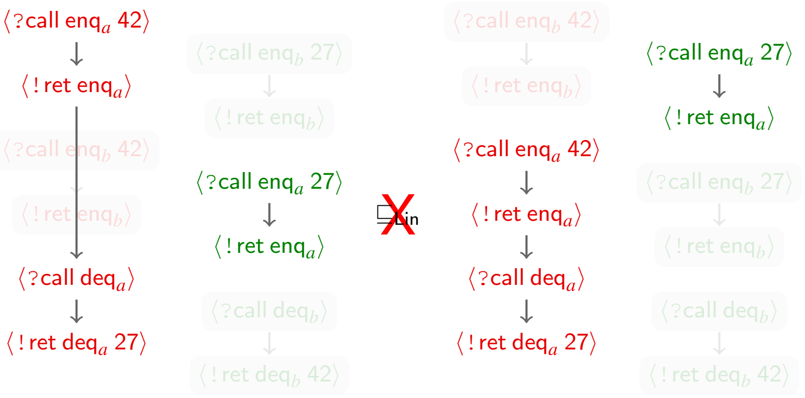


Crucial point:

$$\{\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \sqsubseteq_{\text{Lin}} \left\{ \begin{array}{l} \langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\langle ?\ g\rangle\langle !\ g\rangle \end{array} \right\}$$

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:



By our definition:

$$\{\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \quad \sqsubseteq_{\mathscr{W}} \quad \{\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\}$$

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:

$\langle ?\text{call enq}_a\ 42\rangle \downarrow \langle !\text{ret enq}_a\rangle$
$\langle ?\text{call enq}_b\ 27\rangle \downarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call enq}_b\ 42\rangle \downarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call enq}_a\ 27\rangle \downarrow \langle !\text{ret enq}_a\rangle$
$\downarrow \langle ?\text{call deq}_a\rangle \downarrow \langle !\text{ret deq}_a\ 27\rangle$
$\langle ?\text{call deq}_b\rangle \downarrow \langle !\text{ret deq}_b\ 42\rangle$

$\not\sqsubseteq_{\text{Lin}}$

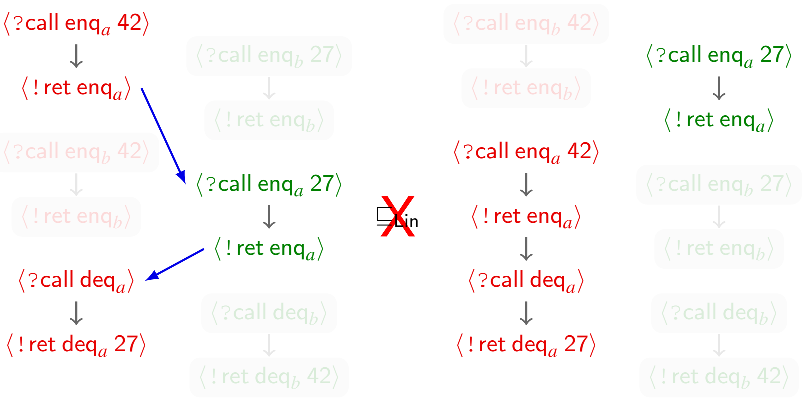
$\langle ?\text{call enq}_b\ 42\rangle \downarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call enq}_a\ 27\rangle \downarrow \langle !\text{ret enq}_a\rangle$
$\langle ?\text{call enq}_a\ 42\rangle \downarrow \langle !\text{ret enq}_a\rangle \downarrow \langle ?\text{call deq}_a\rangle \downarrow \langle !\text{ret deq}_a\ 27\rangle$
$\langle ?\text{call enq}_b\ 27\rangle \downarrow \langle !\text{ret enq}_b\rangle$
$\langle ?\text{call deq}_b\rangle \downarrow \langle !\text{ret deq}_b\ 42\rangle$

By our definition:

$$\{\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \sqsubseteq_{\mathscr{W}} \left\{ \begin{array}{l} \langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\langle ?\ g\rangle\langle !\ g\rangle \end{array} \right\}$$

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:

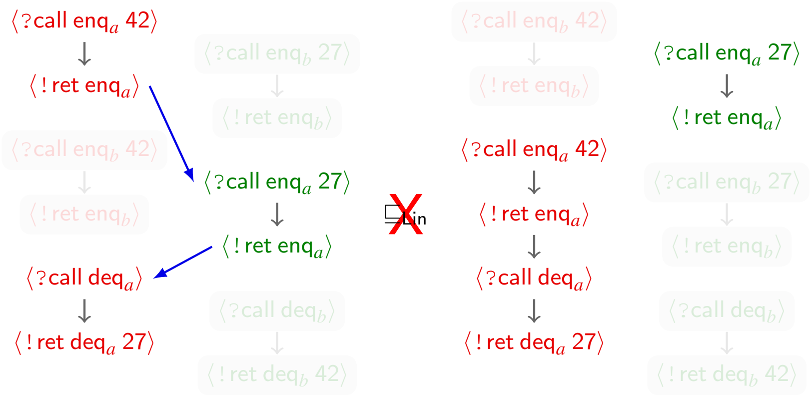


By our definition:

$$\{\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \sqsubseteq_{\mathscr{W}} \left\{ \begin{array}{l} \langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\langle ?\ g\rangle\langle !\ g\rangle \end{array} \right\}$$

Compositionality counterexample (Herlihy/Wing)

Non-Linearizable trace:



In the absence of the *happens-before* edge $\langle !\ g\rangle \longrightarrow \langle ?\ f\rangle$, is this:

$$\{\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\}$$

a reasonable spec?

“Accidental” versus “essential” order

- Same under strong memory, not weak
- Two paths to compositionality
  - Ban accidental order in specs (closure property)
  - Reintroduce global clock
- Second path may be the only path in language with “weak” cas
  (weak cas = synchronization w/o happens-before)

$$\{\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\} \text{ versus } \left\{ \begin{array}{l} \langle ?\ g\rangle\langle !\ g\rangle\langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ g\rangle\langle !\ g\rangle\langle ?\ h\rangle\langle !\ h\rangle \\ \langle ?\ f\rangle\langle !\ f\rangle\langle ?\ h\rangle\langle !\ h\rangle\langle ?\ g\rangle\langle !\ g\rangle \end{array} \right\}$$

Summary

- Compositional reasoning, relaxed memory
- Linearizability using trace sets and happens-before
  - Refinement theorem
  - For strong, tso, pso and jmm
  - Symmetry between “client” and “library”
- Definition of “local” sequential consistency (and local DRF)
  - Refinement theorem for special cases using $\sqsubseteq_{\mathsf{strong}}$ to verify client

(For online version, google “James Riely”)