Linearizability

Past Weeks

- Traces
- Correctness → Constrain Admissible Traces
  - Soundness
  - Liveness
- RPC:
  Request ⟷ Response
- Today: Linearizability: A specific correctness condition

Linearizability Is Really About Concurrency

↳ "How Should A Data Type Behave With Concurrent Accesses"

- Does not consider failures, Losses, Etc.
- Why Interesting?

Abstract Data Type

Some Object Queue Stack . . .

With Operations Enqueue, Dequeue Push Pop . . .

And A Sequential Specification

Behavior when A Single Process Issues One Request At A Time.

Queues: FIFO

P_0 q.enq(1)

ok

Stack: LIFO

P_0 s.push(1)

ok

P_0 q.enq(2)
OK
P_0 q.deq()
OK(1)

P_0 s.push(2)
OK
P_0 s.pop()
OK(2)

P_0 q.deq()

P_0 s.pop()

Main Takeaway

Tells Us Behavior When A Single Process Performs One Operation At A Time

Consistency In This Lecture

Given A Partially Ordered Set Of Concurrent Operations Specify How ADT Behaves

↳ Equivalent Sequential History

Linearizability

Partial Order $<_H$

Equivalent Seq. History S $\supseteq$

$\boxed{<_S \cancel{\subseteq} <_H}$

& Correct Seq. Specification

Here lie errors & dragons.

Sela, Herlihy, Petrank PODC '21

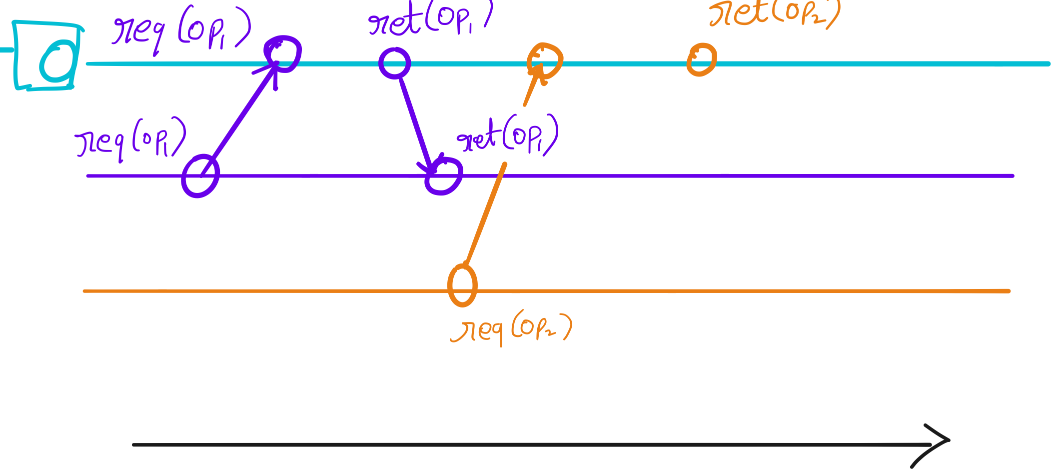
$<_H$: Partial order on operations

$op_1 <_H op_2$

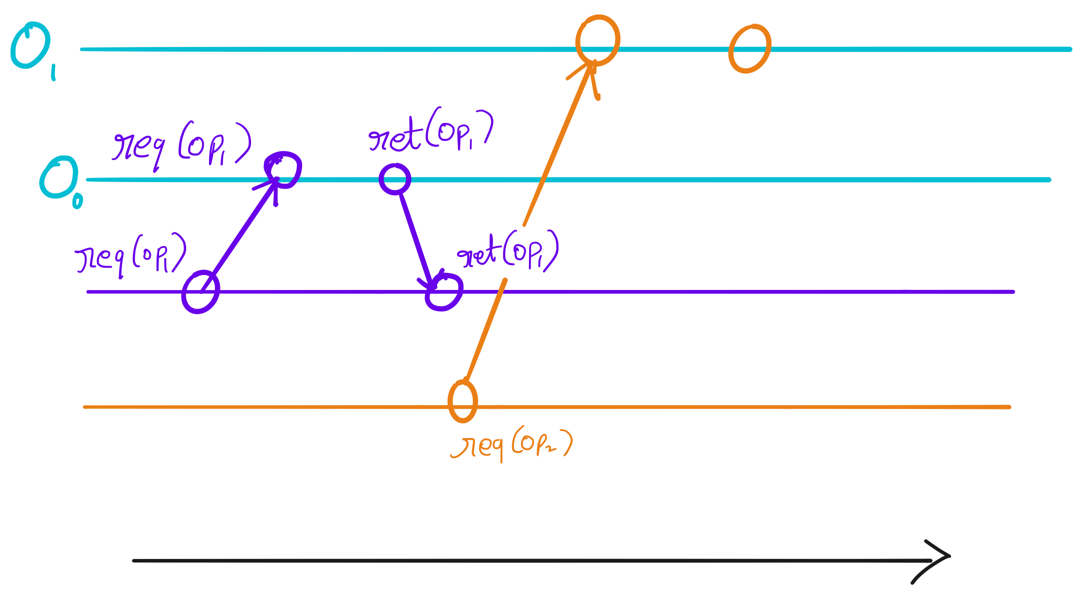
$req(op_1)$ $ret(op_1)$ $req(op_2)$ $ret(op_2)$

Before

↳ In Whose Frame Of Reference / According To Whom?



Might really be a group of processes rather than one.



Some object O is linearizable implementation of ADT

$\Longrightarrow$

Given H, $<_H$ can extend H to H' so

$\exists$ sequential history S ($<_S$) compliant with ADT

s.t.

① complete(H') is equivalent to S

① complete(H) ...

② $<_H \subseteq <_S$

Walk through this step by step

① complete(H): Maximal subsequence of H where every req has a response

H: ⟨req, op_1, P_0⟩

complete(H) =

H: ⟨req, op_1, P_0⟩ ⟨ret, op_1, P_0⟩

complete(H) =

② Extend H to H'

↳ Can add missing responses/rets.

→ Necessary to deal with operations that have visible effect before response

req O_1 req O_2 ret O_2

H_0: ⟨req, enq(2), P_0⟩ ⟨req, deq(), P_1⟩ ⟨ret, OK(2), P_1⟩

H_0': req O_1 req O_2 ret O_2 ret O_1

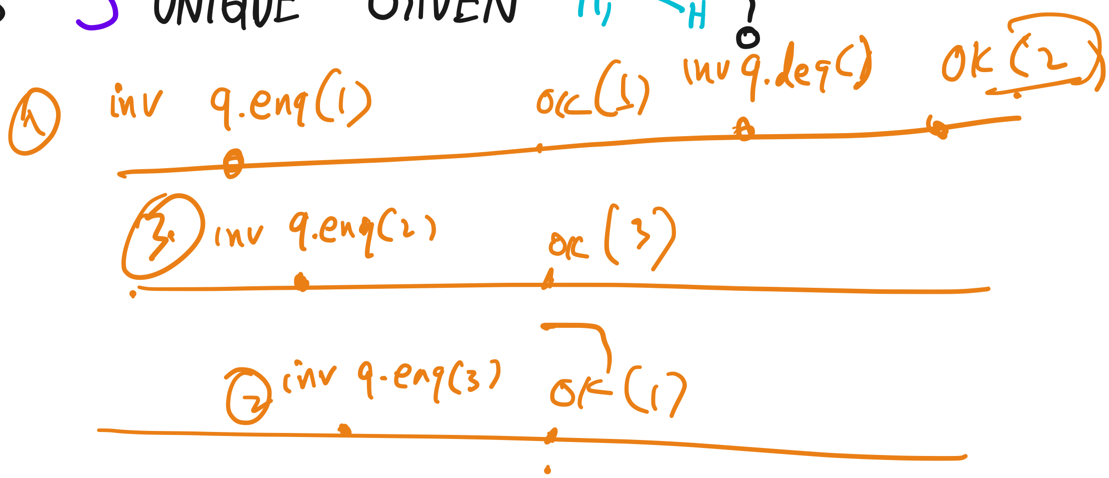
③ Find S, $<_S$

... : { }

$S \quad O_1 \quad O_2$

$<_S$

Q1: Is S UNIQUE GIVEN H, $<_H$?



$$D, \quad H, <_H \longrightarrow S$$

Q2 We produce $S, <_S$ equivalent to complete(H'), but only require $<_H \subseteq <_S$. Is that Ok?

Core problem: $<_H$ only orders "operations" in H

Both invocation (req) & response (ret) in H

"Proof by citation :|"

As written some operations can be reordered

$H'|_o$: $\underbrace{\langle req, deq(\,), p_0\rangle}_{req(op_1)}$ $\underbrace{\langle ret, ok(1), p_0\rangle}_{ret(op_1)}$ $\underbrace{\langle req, enq(\,), p_1\rangle}_{req(op_2)}$

$H'|_o$: $req(op_1)\ ret(op_1)\ req(op_2)\ ret(op_2)$

$<_H : \{\}$

$<_{H'} : \{(op_1, op_2)\}$

Allows locality to be broken

$S|_o : [op_2 ; op_1]$

$<_H \subset <_S$

$<_{H'} \not\subset <_S$

Herlihy's correction [PODC 2021]

Should be

$<_{complete(H')} \subseteq <_S$

ctr.set(2) ok(1) ; ctr.set(1)

LOCALITY

- If a system has multiple objects $O_1, O_2, \cdots O_n$ the system as a whole is linearizable $\iff$ $O_1, O_2, \cdots O_n$ linearizable

H linearizable $\iff$ $H|_{O_i}$ linearizable

$\llcorner$ Only operations on O_i

H linearizable $\Rightarrow$ $H|_{O_i}$ Linearizable

"Easy direction"

- Observation

$$\begin{array}{ccc} H, & H', & \text{complete}(H') \\ \sqcup & \sqcup & \sqcup \\ H|O_i & H'|O_i & \text{complete}(H'|O_i) \end{array}$$

$$<_{H|O_i} \subseteq <_H$$

$\Rightarrow$ $<_{H|O_i} \subseteq <_S$ & easy to see $<_{H|O_i} \subseteq <_{S|O_i}$

Also S equivalent to complete(H') $\Rightarrow$
$S|O_i$ equivalent to Complete$(H'|O_i)$

S compliant with $O_1, \dots O_n$ $\Rightarrow$
$S|O_i$ compliant with O_i

$H|O_i$ linearizable $\forall$ O_i $\Rightarrow$ H linearizable

- More complicated. <u>Why care?</u>
- Instructive to see what it means when this is not true.

Two examples

① Broken linearizability definition

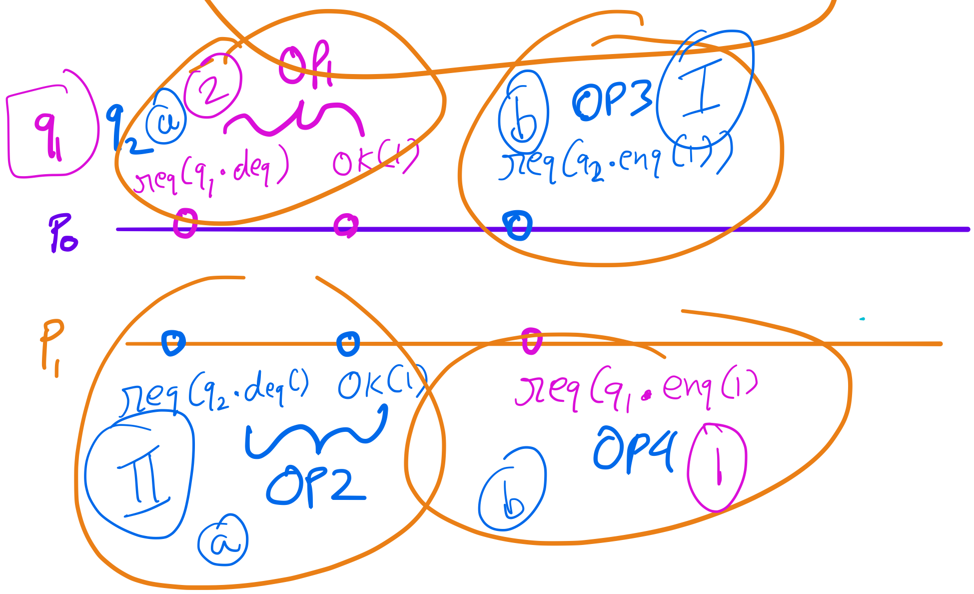
① $<_H$ only defined on operations with request & response

② Require $<_H \subseteq <_S$

H|o_i

H|o_i lin ⇒

H lin ✗



$<_H$: { }

[Note: Sequential history requires maintaining process order ⇒ Op3 req must be after op1 ret]

H|q_1 can be linearized

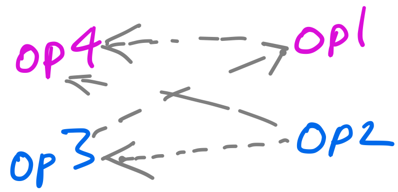
op_4 ; op1 [Same ~~bug~~ way as before]

H|q_2 can be linearized

OP_3 ; OP_2

H cannot be linearized

cannot



Why is this not a problem with the fix?

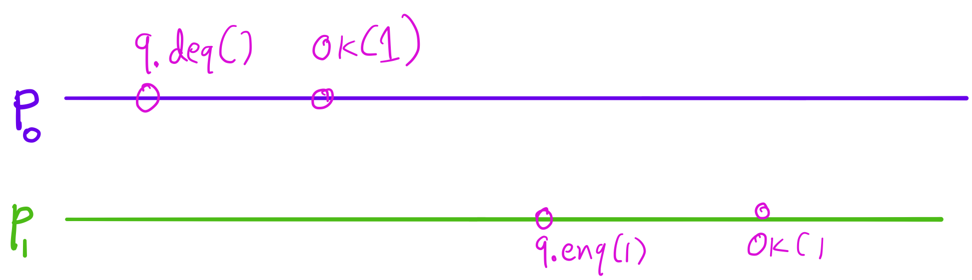
② Sequential Consistency

- Different model.

For process $P_0, P_1, P_2, \ldots P_n$ requires finding $S, <_s$ s.t.

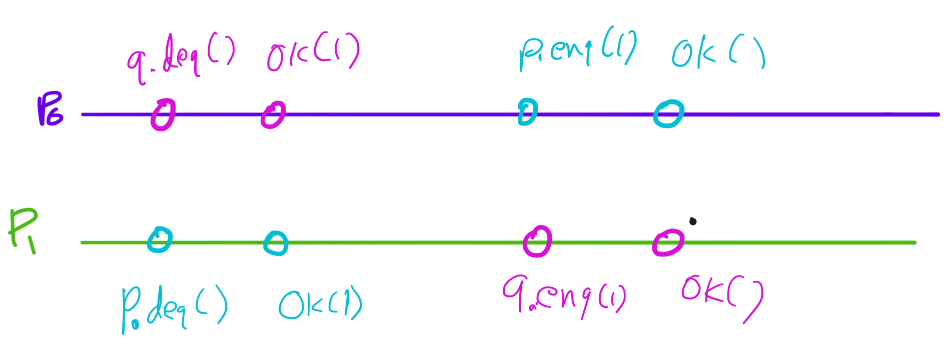
$$<_{p_i} \subseteq <_s$$

"No real-time ordering required



Seq Cst ✓

Linearizability ✗



$H|q$ is Seq cst (see above)

$H|p$ " " " (symmetry)

H is NOT Seq Cst

$H|O_i$ linearizable $\forall\, O_i \Rightarrow H$ linearizable

<u>Sketch</u>

$\forall O_i$ $H|O_i$ linearizable

$\Rightarrow$ Given $H|O_i$ can find $H_i' = H|O_i + R_i$

s.t. $\exists$ sequential history S_i where

S_i equivalent to H_i'

& $<_{complete(H_i')} \subseteq <_{S_i}$

Proof by construction

$$H' = H + \sum_i R_i$$

$$<_{H'} = \boxed{<_H} \cup \left(\bigcup_i <_{H_i'}\right) \cup \boxed{\text{Transitive edges}}$$

if $e_1 <_{H'} e_2 \wedge e_2 <_{H'} e_3$ then $e_1 <_{H'} e_3$

$<_{H'}$ Is A PARTIAL ORDER

↳ IRREFLEXIVE $(\,(e,e) \notin <_{H'})$ By construction

$(a, b) \in S$

→ ~~Transitive~~ BY CONSTRUCTION

→ Antisymmetric

$(a,b)\ (b,c) \in S$ $\Rightarrow$ Antisymmetric

$\Rightarrow (a,c) \in S$

↳ No cycles s.t.

$$e_1 <_{H'} e_2 <_{H'} e_3 \dots <_{H'} e_1$$

$(a,b)\ (b,c)\ (a,c)$

(c,a)

Proof by <u>contradiction</u>

Assume Otherwise

(I) $e_1, e_2, \dots, e_n$ are not all operations on the same object

$<_{H_i'}$ partial order $\forall i$

(II) Pick smallest cycle s.t. e_1, e_2 are ops on $o_1 \neq o_2$

Lemma None of $e_2, \dots e_n$ are ops on o_1

↳ Assume otherwise, e_i is first op after e_1 on o_1

$$e_2 <_H e_i \Rightarrow Ret(e_2) \text{ Before } Inv(e_i)$$

$$e_1 <_H e_2 \Rightarrow Ret(e_1) \text{ Before } Inv(e_2)$$

$\Rightarrow$ By construction

$$e_1 <_{H'} e_i$$

$\Rightarrow e_1, \cancel{e_i}, e_{i+1}, \dots e_n$ is

Smaller cycle

Contradiction

$\Rightarrow$ e_1 & e_n are ops on different objects

$\Rightarrow$ $e_1 <_H e_2$; $e_n <_H e_1$

But $<_H$ is transitive

$\Rightarrow$ $e_n <_H e_2$

$\Rightarrow$ $e_2, \ldots e_n$ is smaller cycle.

No cycle exists

CLAIM Can construct sequential history S consistent with complete(H') where $<_{complete(H')} \subseteq <_S$

Non-Blocking

↳ Why useful?

$op1 <_H op2$

$<_I$: $op_1 <_I op_2 \Leftrightarrow$ inv(op_1) before inv(op_2)

$<_I \subseteq <_{S'}$

$<_H$

→ Why true

CAP theorem

Consistency : Linearizability

Availability

Partition tolerance :