

Distributed Systems

Lecture 3

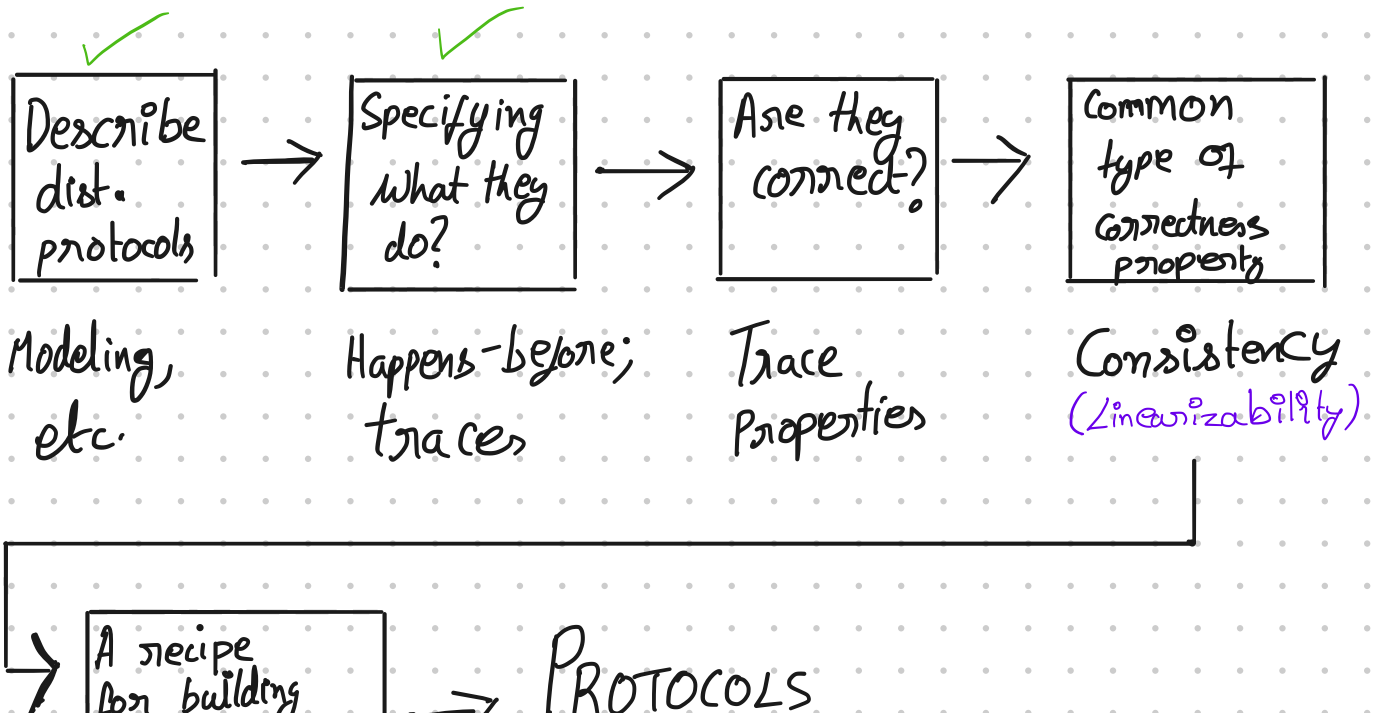
Linearizability

Announcement: THE SCHEDULE (not the notes) ARE AUTHORITY

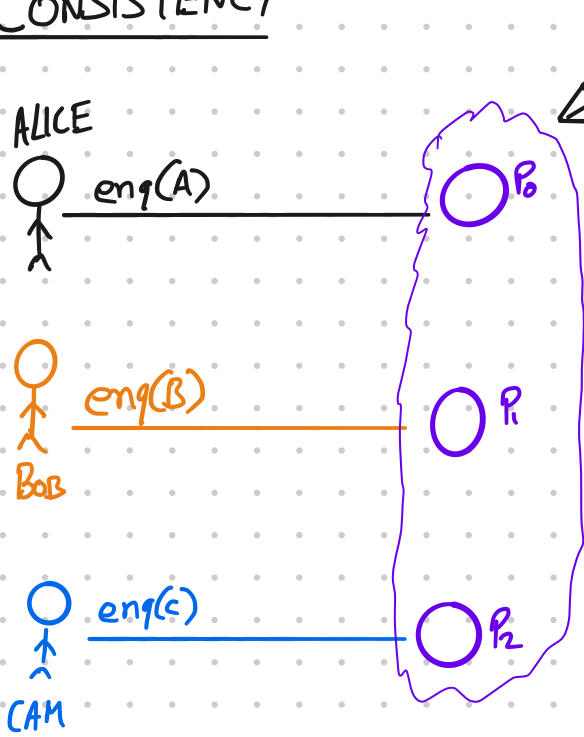
Why trace properties; happens before; etc.

- Came up in last class.
- Today's topic might raise similar questions
- Added a note motivating trace properties but...

Class structure



CONSISTENCY



One system implementing a queue.

\Downarrow
 ABC
 CAB
...
In what order do operations get performed?

Note: What consistency model is a choice made by the protocol designer/distributed system builder!! A correctness requirement
- Safety or Liveness?

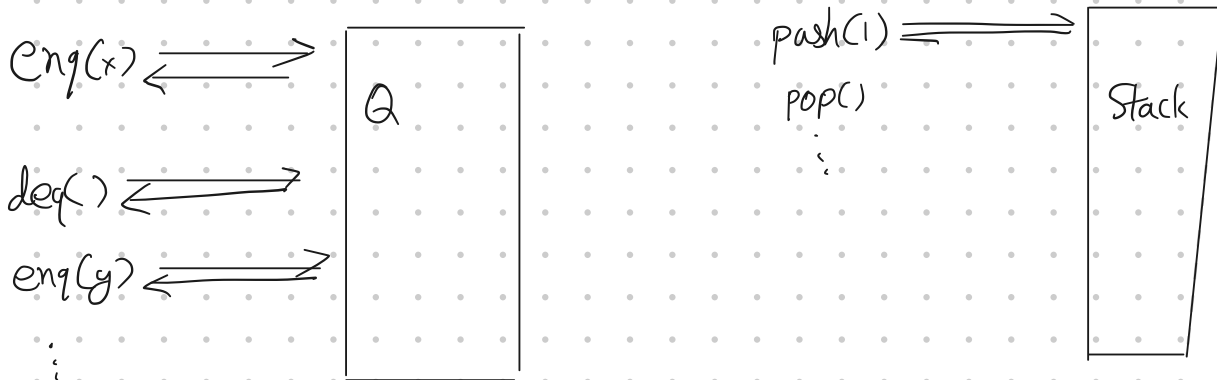
When making the choice, need to balance (TRADEOFF BETWEEN)

- Ease of use
 - ↳ Do programmers need to be extra cautious when using
 - Do users understand the behavior
- Performance } Tied to each other.

- Fault tolerance

How to think about an ABSTRACT DATA TYPE

Q1. Why is there a DATA TYPE in this DISTRIBUTED Sys. class?



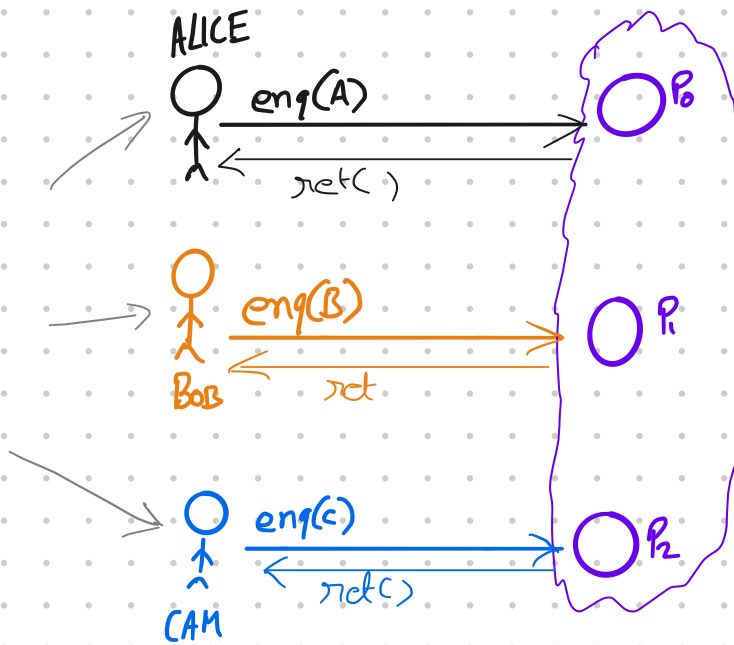
Q2. What is so abstract about this?

NIR SHAVITS ~ 2011:
(Older wisdom)

...infinitely easier and more intuitive for us humans to specify how abstract data structures behave in a sequential setting, where there are no interleavings.

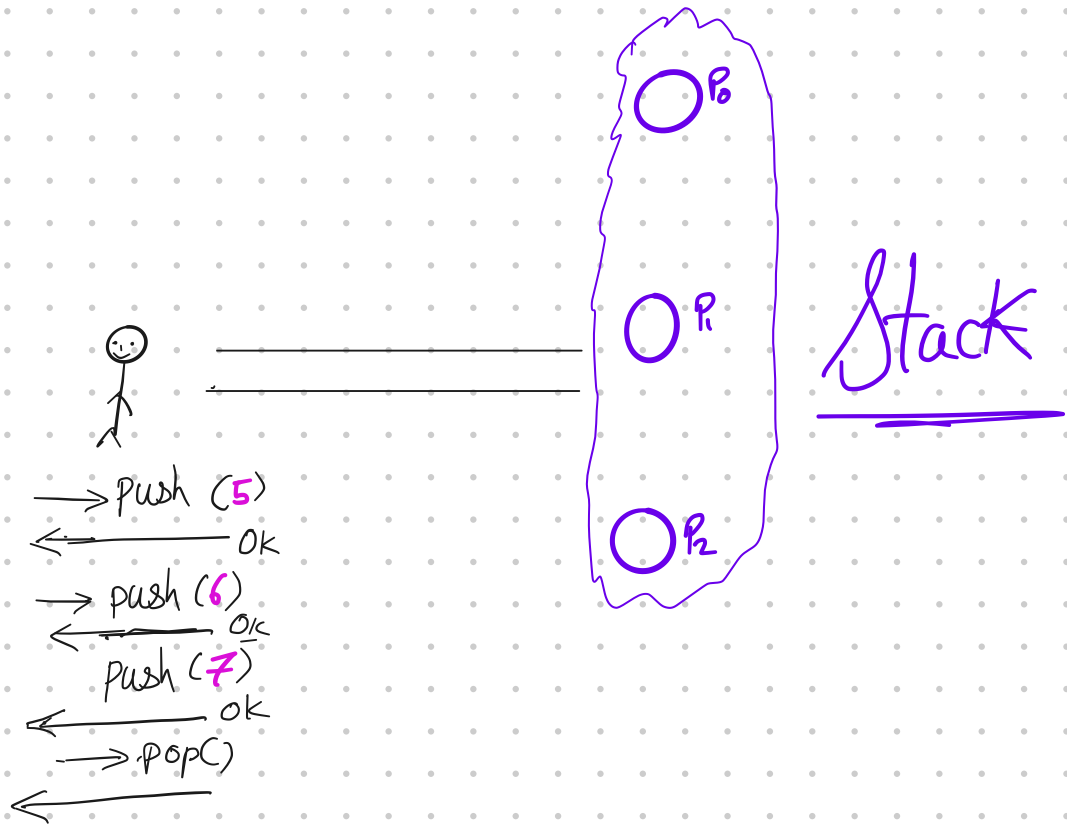
Specify sequential Behavior AND Mapping from concurrent to sequential.

Note: What we are focused on what CONCURRENT USERS observe.



Sequential Specification

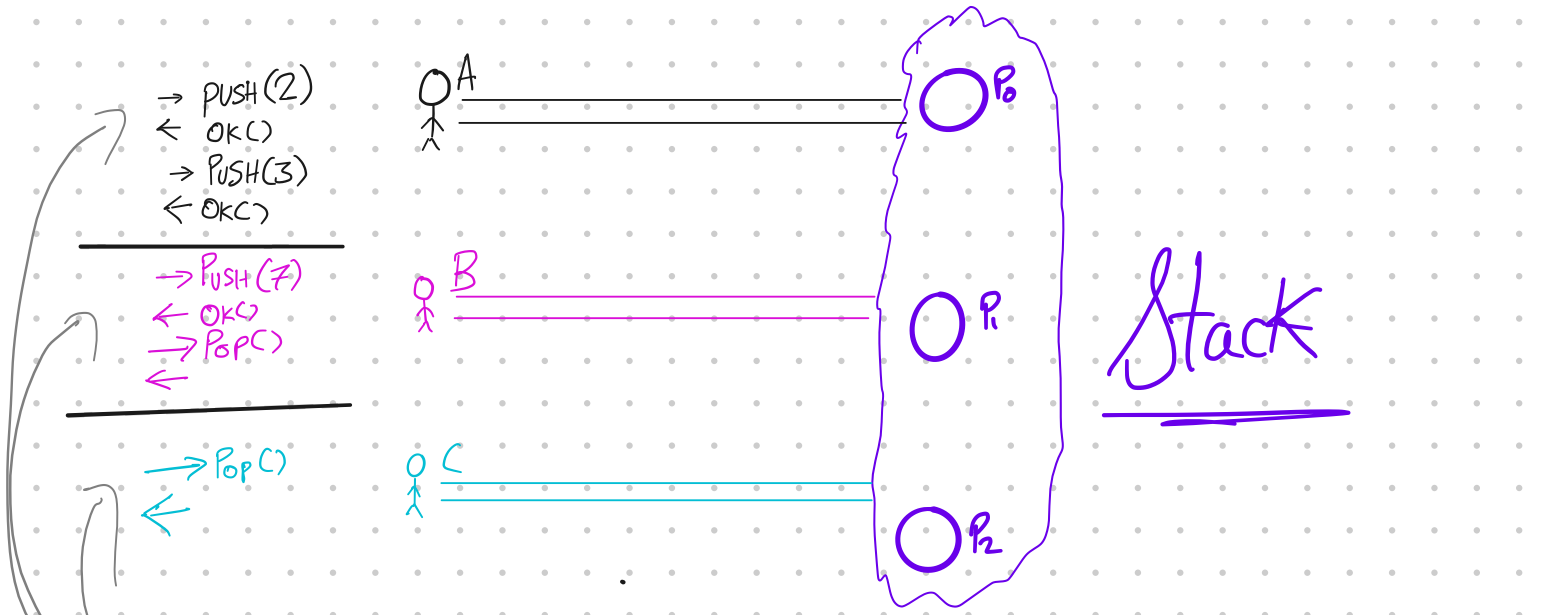
- Behavior without concurrency



- ① Exactly one operation at a time.
- ② Wait for response before issuing next operation
- ③ Says what each operation can return.

What we are used to: Queues are FIFO, stacks are LIFO, Text editors ...

Mapping from Concurrent Execution



PROGRAMMER

① Operations complete between invocation ($\text{inv}, \rightarrow, \dots$) and response (resp, \leftarrow)

- ② \exists a total order that
- Ⓐ All users can agree on
 - Ⓑ Is consistent with seq. specification

Linearizability

Now more formally

(a) Notation from paper

object operation process : Invocation

object OK(value) process : Return

Remember, a process has at most 1 operation in progress at a time.

(b) $<_H$ PARTIAL ORDER

$op_1 <_H op_2 \Rightarrow$ Return op_1 before invocation op_2

Operations complete between invocation (inv, \rightarrow, \dots) and response ($\leftarrow, resp$)

\hookrightarrow op_2 should observe effects from op_1

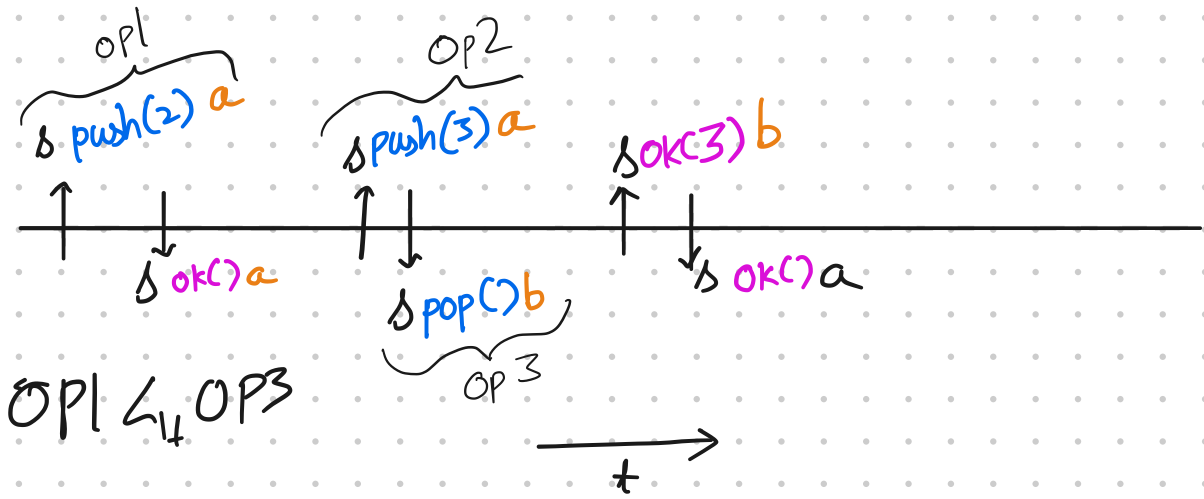
But remember, the total order(s) we will look for are SUPERSETS of $<_H$

Observe

$<_H$ orders all operations issued by

the same process.

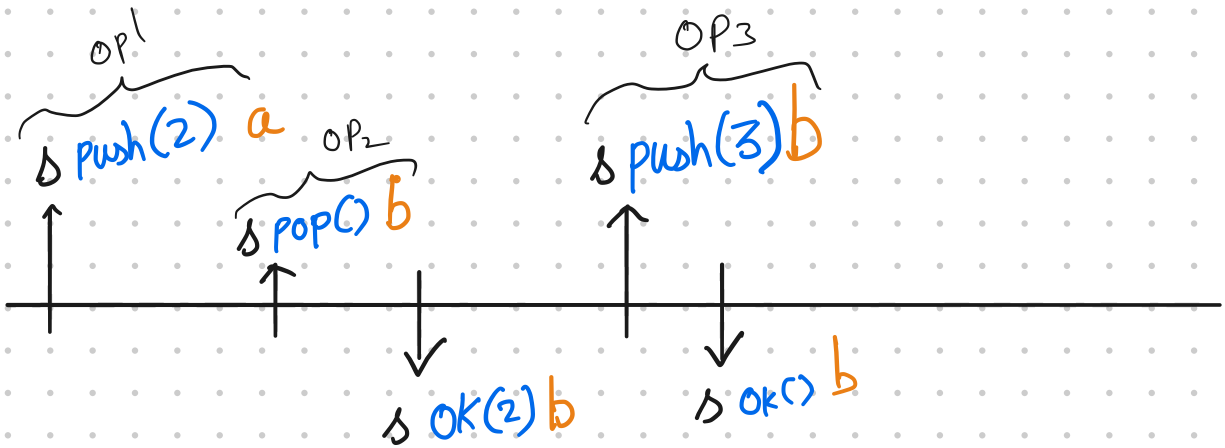
Why?



OP1 \prec_4 OP2 OP3

?

?



OP1 OP2 \prec_{lt} OP3

© History H linearizable if and only if

... more

$\exists H$ that extends H (by \emptyset on \dots)

returns) s.t.

there is total order

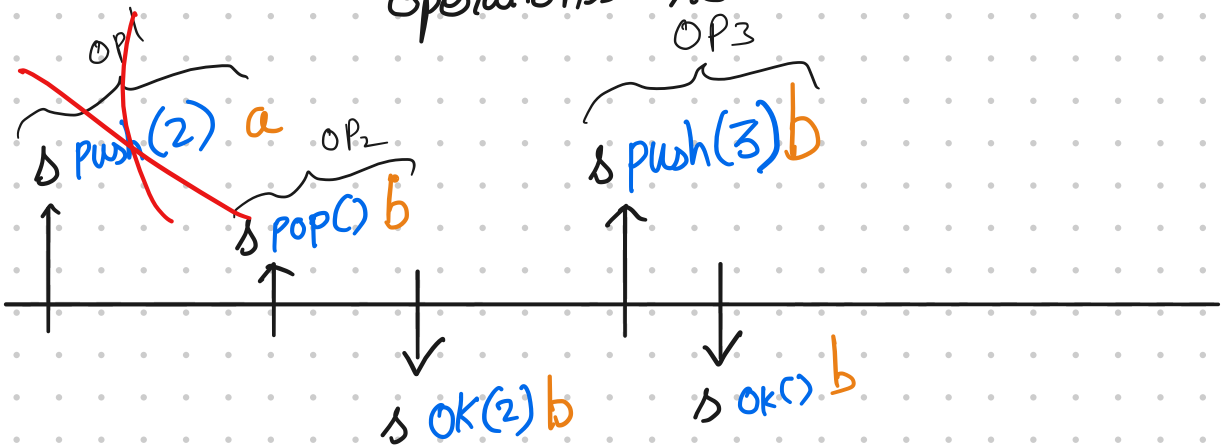
\prec_s where $\prec_{\text{complete}(H)} \subseteq \prec_s$

$\Delta \prec_s$ meets SEQUENTIAL SPEC.

- complete (H) : H with all incomplete / inprogress

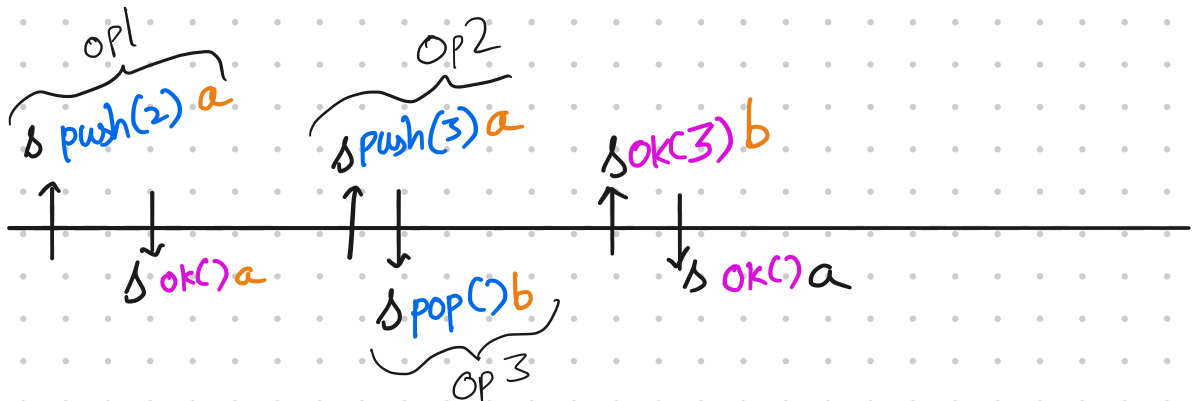
operations removed

$H \equiv$



Operations in COMPLETE (H) ?

$H \equiv$



Operations in COMPLETE (H) ?

to it and only it

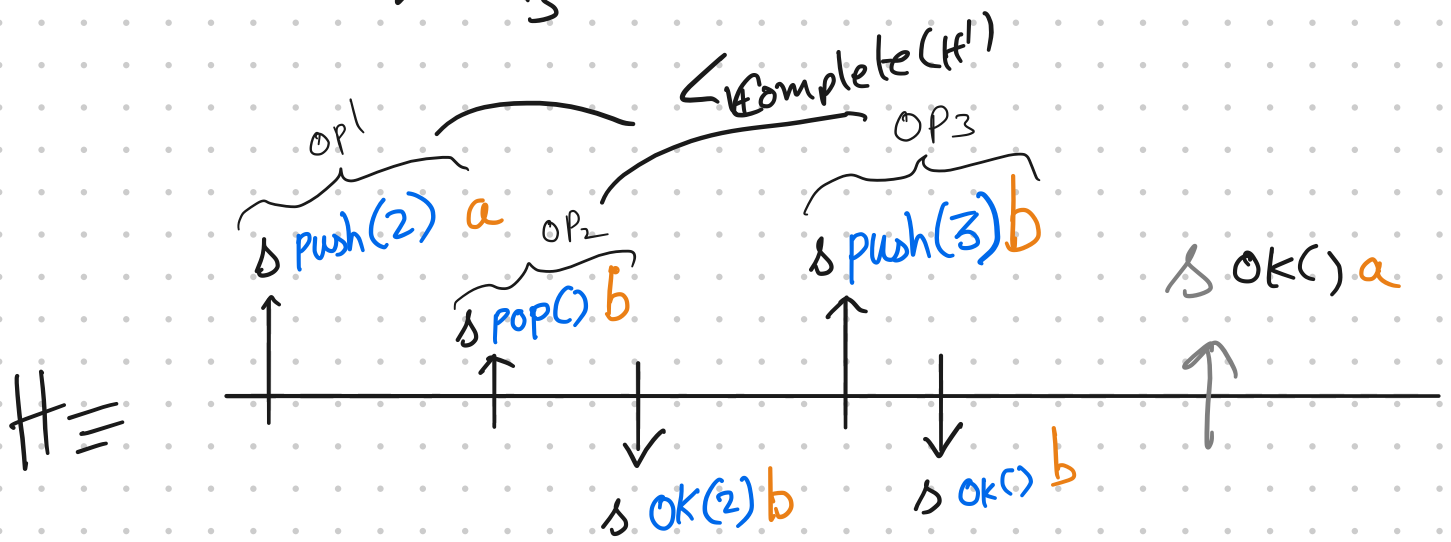
$\exists H'$ that extends H (by \emptyset or more

returns) s.t.

there is total order

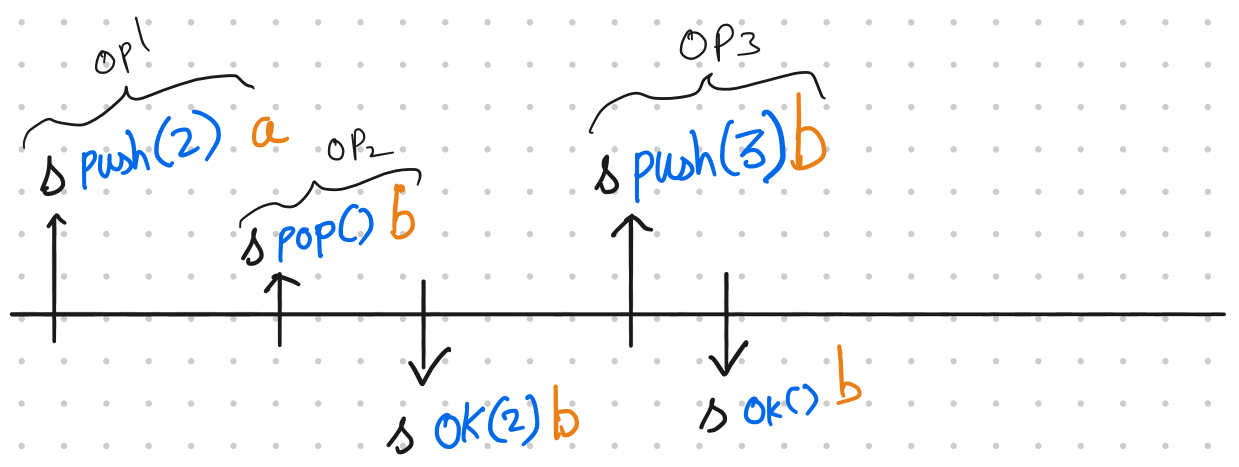
\prec_s where $\prec_{\text{complete}(H')} \subseteq \prec_s$

$\triangleright \prec_s$ meets SEQUENTIAL SPEC

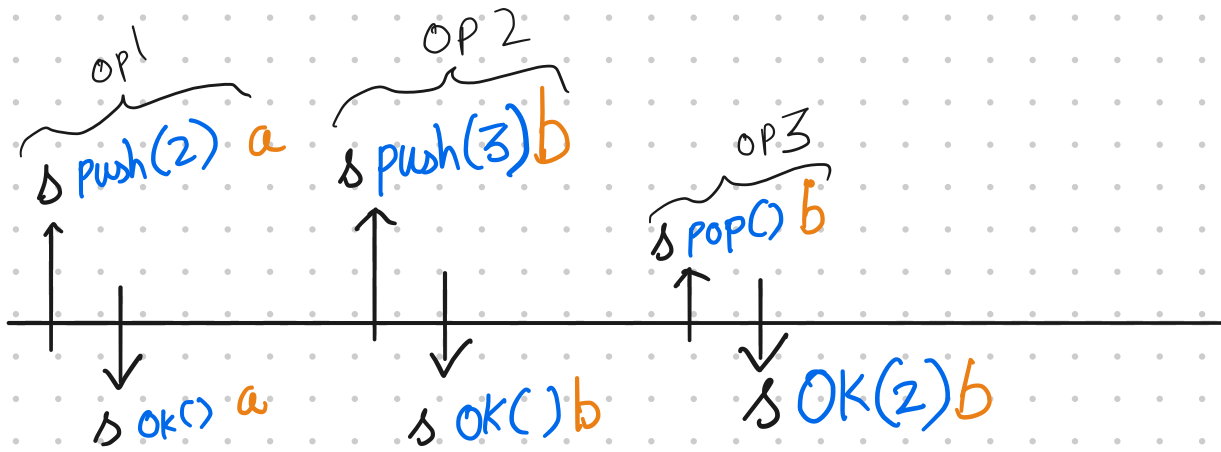
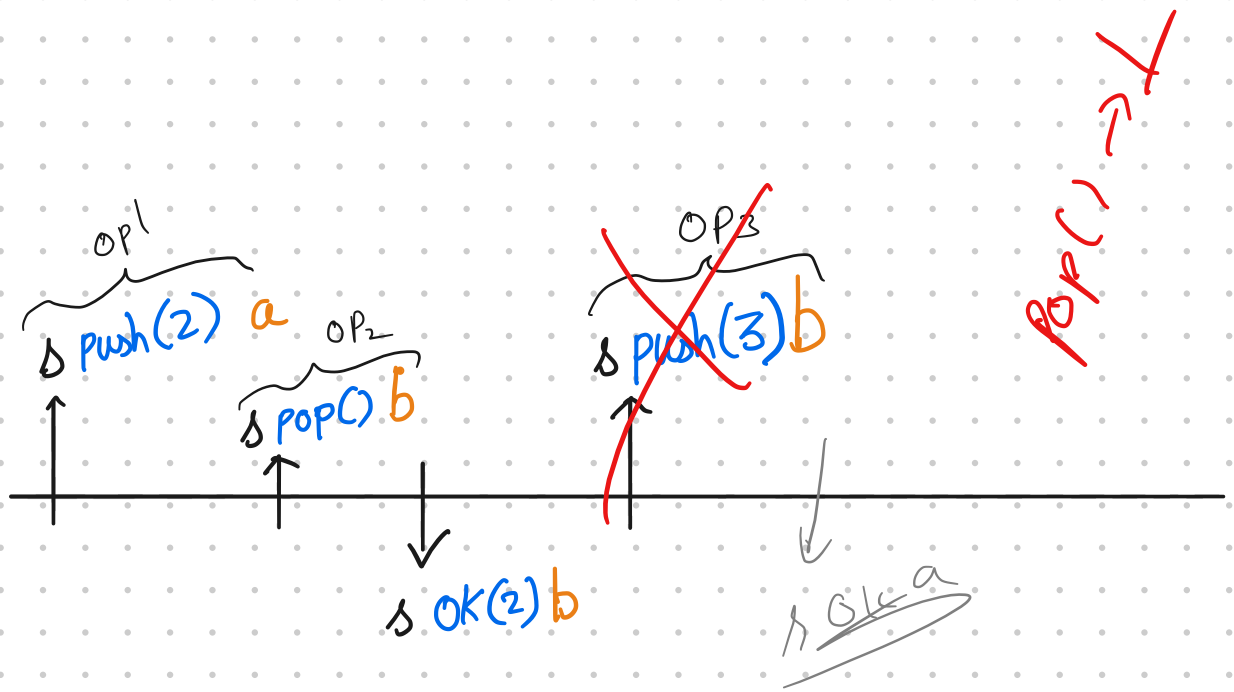


$\prec_s = OP_1 \prec_s OP_2 \prec_s OP_3$

Is H a linearizable history for a stack?



$H \rightarrow H'$ $H' \rightarrow \text{Complete}(H')$ $\leftarrow s$



OK. BUT ARE THERE OTHER OPTIONS?

Seq. consistency

~~① Operations complete between invocation ($inv_i \rightarrow r_i$) and response ($r_i \leftarrow$)~~

② \exists a total order that

(a) All users can agree on

(b) Is consistent with seq. specification

\langle_p : Order of ops from process p

Equivalently $\langle_{H|P}$ [$H|P$: Only invocations & returns for process p]

History H seq. w.r.t. ϕ if and only if

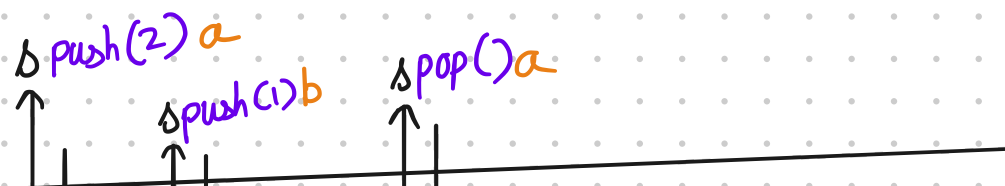
$\exists H'$ that extends H (by ϕ or more returns) s.t.

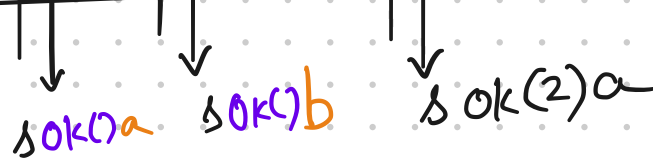
there is total order

\langle_s where $\langle_{\text{complete}(H'|P)} \subseteq \langle_s \forall p$

$\Downarrow \langle_s$ meets SEQUENTIAL SPEC

- Common: Strongest guarantee provided by C++, Rust, etc.
- Might appear that operations move in time





- Linearizable?

- Seq cst?

Properties

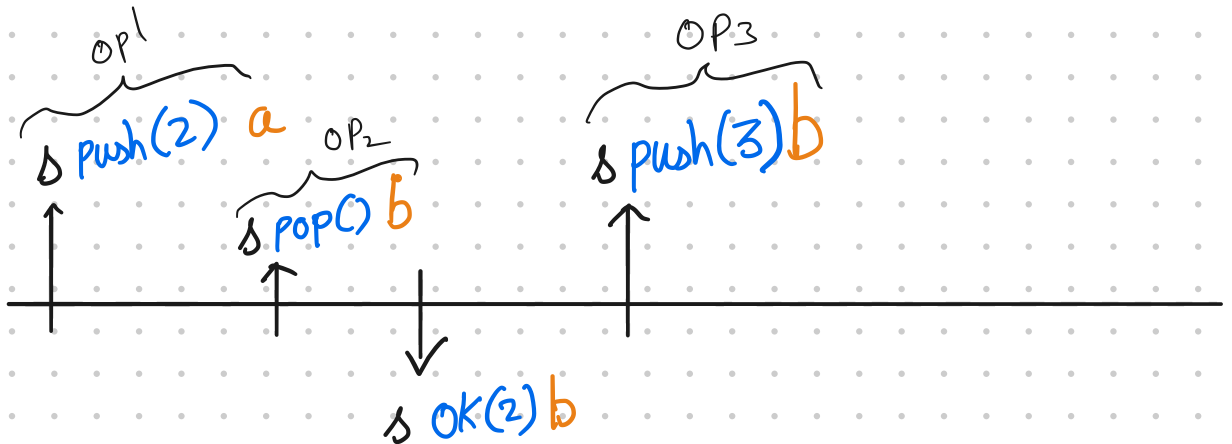
Non-Blocking

If history H is linearizable and

H contains incomplete op O then

\exists response O_R to O such that

$H; O_R$ is linearizable

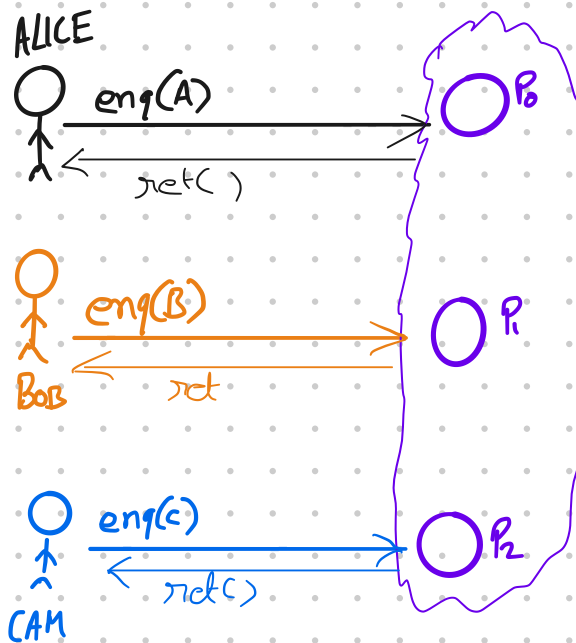


Two incomplete operations: op_1, op_3

D.

Why is this useful?

- Can delay operation execution without violating consistency



Local

H has operations for two objects
x and y

x pop() a
y push() a
y pop() b

H linearizable

↔

H | x linearizable &
H | y linearizable

What this means?

- The linearization of H (S or \leftarrow_S) matches x 's seq spec \triangleright y 's seq spec.

- Cannot say anything about a data structure with a different seq. spec that

invoke push(2) a
invoke push(3) b
invoke get-len() c
return OK(1) c

combines x & y

struct stack-with-len Σ
stack x
int len

push
pop
get-length

}

- Why useful?

Begin Txn

a

b

c

End Txn

Begin Txn

c

d

End Txn



$(x$

\mathbb{Z}^2

$$(x_0, x_1) < (y_0, y_1)$$

$$\text{iff } \begin{array}{l} x_0 < y_0 \\ x_1 < y_1 \end{array}$$

$$(1, 2) < (3, 4)$$

$$(1, 2) \quad (2, 1)$$