LINEARIZABILITY
"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, Push...
Dequeue Pop

And a **sequential specification**

Behavior when a single process issues one request at a time.

Queues: FIFO
- `p, q.enq(1)`
  - `ok`<

Stack: LIFO
- `p, s.push(1)`
  - `ok`<
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.
Linearizability

Partial Order $\prec_H$

Equivalent Seq. History $\approx$

Correct Seq. Specification

Here lie errors
dragons.
Sela, Hershly, Pentmak
PODC '21

$\prec_H$ Partial order on operations

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

$\Rightarrow$

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

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

$s.t.

\begin{itemize}
\item complete($H'$) is equivalent to $S$
\end{itemize}
Walk through this step by step

1. \text{complete}(H) : Maximal subsequence of \( H \) where every \( \text{req} \) has a response

\[
H : \langle \text{req}, \text{op}_1, p_0 \rangle
\]

\[
\text{complete}(H) =
\]

\[
H : \langle \text{req}, \text{op}_1, p_0 \rangle \langle \text{net}, \text{op}_1, p_0 \rangle
\]

\[
\text{complete}(H) =
\]

2. Extend \( H \) to \( H' \)

\( \rightarrow \) Can add missing responses/\( \text{net} \)'s.

\( \rightarrow \) Necessary to deal with operations that have visible effect before response.

\[
H_0 \quad \langle \text{req}, \text{eq}(2), p_0 \rangle \quad \langle \text{req}, \text{deq}(1), p_1 \rangle \quad \langle \text{net}, \text{ok}(2), p_1 \rangle
\]

\[
H_0 \quad \text{req} \quad \text{eq}_0 \quad \text{req}\_0 \quad \text{net} \quad \text{net}^2
\]

3. Find \( S \), \( <_S \)

\[
<_S
\]
Q1: Is $S$ unique given $H$, $<_H$?

\[ \text{inv q.enq}(1) \quad \text{ok}(3) \quad \text{inv q.deq}(1) \quad \text{ok}(2) \]

\[ \text{inv q.enq}(2) \quad \text{ok}(3) \]

\[ \text{inv q.enq}(3) \quad \text{ok}(1) \]

$D$, $H$, $<_H \rightarrow S$

---

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

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

"Proof by citation ↓"  

\[ \text{Both invocation (req)} \quad \text{response (net)} \]

As written, some operations can be reordered.
$H_0 \leq \langle \text{neq, deq, lo} \rangle \leq \langle \text{net, ok} \rangle \leq \langle \text{neq, eq, ho} \rangle \leq \langle \text{net, op} \rangle \leq \langle \text{neq, ho} \rangle$

$H_0$ with $	ext{neq (op), net (op), neq (op), net (op)}$

$\preceq_H : \exists \exists$

$\preceq_{H'} : \exists (\text{op}_1, \text{op}_2) \exists$

Allows locality to be broken

S: $\exists \text{op}_2; \text{op}_1$ $\preceq_H$ $\subset_s$

Henthys's connection [PODC 2021]

Should be

$\text{complete}(H') \preceq \subset_s$

Locality

If a system has multiple objects $o_1, o_2, \ldots, o_n$

the system as a whole is linearizable

$\iff o_1, o_2, \ldots, o_n$ linearizable

$H$ linearizable $\iff H|_{o_i}$ linearizable
H linearizable $\Rightarrow$ $\exists l_0$: linearizable

"Easy direction"

- Observation

H, H', complete (H')
U U U U
H l_0: H' l_0: complete (H' l_0)

$\preceq_h l_0 \subseteq \prec_h$

$\Rightarrow$ $\preceq_h l_0 \subseteq \prec_s$ & easy to see $\prec_h l_0 \subseteq \prec s_{l_0}$

Also $S$ equivalent to complete (H') $\Rightarrow$
S l_0: equivalent to complete (H' l_0)

S compliant with $o_1, \ldots$ on $\Rightarrow$
S l_0: compliant with $o_1$

H l_0: linearizable $\forall o_i \Rightarrow$ H linearizable

- More complicated: Why care?

- Instructive to see what it means when this is not true.
Two examples

1. Broken linearizability definition
   (1) \( \preceq_H \) only defined on operations with request & response
   (2) Require \( \preceq_H \leq \preceq \)

\[ H_0: \text{lin} \rightarrow H_{\text{lin}} \times \]

\[ \preceq_H: \text{lin} \]

\[ H_{\text{lin}} \times \]

\[ \text{[Note: Sequential history requires maintaining process order] \( \Rightarrow \) Op3 req must be after Op1 net] \]

\[ H_{\text{lin}} \text{ can be linearized} \]

\[ \text{Op}_4 \preceq \text{Op}_1 \]

\[ H_{\text{lin}} \text{ can be linearized} \]

\[ \text{Op}_3 \preceq \text{Op}_2 \]

\[ H_{\text{lin}} \text{ cannot be linearized} \]
Why is this not a problem with the fix?

2) Sequential Consistency
   - Different model:
     For processes $P_1, P_2, ..., P_n$, requires finding $S$, such that:
     \[
     \preceq_p \leq \preceq_s
     \]
     "No real-time ordering required."

\[ q_{deq(0)} \text{ OK}(1) \]
\[ P_0 \]

\[ q_{enq(1)} \text{ OK}(1) \]
\[ P_1 \]

Seq Cons ✓

Linearizability ✗
H1q is Seq Cst (see above)

H1p " " " (symmetry)

H is NOT Seq Cst

\[ H1o: \text{linearizable} \quad \forall o; \quad \Rightarrow \quad H \text{ linearizable} \]

Sketch

\[ \forall o; \quad H1o: \text{linearizable} \]

\[ \Rightarrow \quad \text{Given } H1o; \quad \text{an } f \text{ and } H1' = H1o + R; \]

s.t. \[ \exists \text{ sequential history } Si \quad \text{where} \]

\[ Si \text{ equivalent to } H1' \]

\[ \forall \quad \langle \text{complete}(Hi) \rangle \subseteq \langle Si \rangle \]

Proof by construction

\[ H' = H + \sum R; \]

\[ \langle H' \rangle = \langle H \rangle \cup \left( \bigcup_i \langle H_i \rangle \right) \cup \text{Transitive edges} \]

\[ \forall e_1, e_2 \quad \langle H' \rangle \text{ then } e_1 \leq e_2 \]

\[ \langle H' \rangle \text{ is a partial order} \]

\[ (e_1, e_2) \subseteq S \]

\[ \Rightarrow \text{ reflexive \quad by construction} \]

\[ \Rightarrow \text{ transitive \quad by construction} \]

\[ \Rightarrow \text{ antisymmetric} \]
\( (a, b) (b, c) \rightarrow (a, c) \in S \rightarrow \text{Antisymmetric} \)

\( \Rightarrow \) No cycles such that:

\[ e_1 \leq e_2 \leq e_3 \ldots \leq e_i \]

**Proof by contradiction**

**Assume Otherwise**

\( e_1, e_2, \ldots, e_n \) are not all operations on the same object

\( \preceq \) partial order \( \forall i \)

**Pick Smallest Cycle s.t.** \( e_1, e_2 \) are ops on \( o_1 \neq o_2 \)

**Lemma** None of \( e_1, e_2, \ldots, e_n \) are ops on \( o_1 \)

\( \Rightarrow \) Assume otherwise, \( e_i \) is first op after \( e_1 \) on \( o_1 \)

\[ e_2 \preceq e_i \Rightarrow \text{Rot}(e_2) \text{ Before Inv}(e_i) \]

\[ e_1 \preceq e_2 \Rightarrow \text{Rot}(e_1) \text{ Before Inv}(e_2) \]

\( \Rightarrow \) By construction

\[ e_1 \preceq e_i \]

\( \Rightarrow e_1, e_i, e_i, \ldots, e_n \) is a cycle.
Smaller cycle

Contradiction

\[ \Rightarrow e_1 \text{ is ops on different objects} \]

\[ \Rightarrow e_1 \prec_H e_2 \quad ; \quad e_n \prec_H e_1 \]

But \( \prec_H \) is transitive

\[ \Rightarrow e_n \prec_H e_2 \]

\[ \Rightarrow e_2, \ldots, e_n \text{ is smaller cycle.} \]

No cycle exists

Claim: Can construct sequential history \( S \) consistent with \( \text{complete}(H) \)

where \( \leq_{\text{complete}(H)} \subseteq \leq_S \)

Non-Blocking

\( \Rightarrow \text{Why useful?} \)

\[ \leq_i : \text{op}, \leq_i \text{ op_2} \Rightarrow \text{inv(op_2) before} \]

\[ \text{inv(op_2)} \]