

BA: CONCURRENCY-AWARE LINEARIZABILITY

Nir Hemed and Noam Rinetzky
Tel Aviv University

PODC 2014

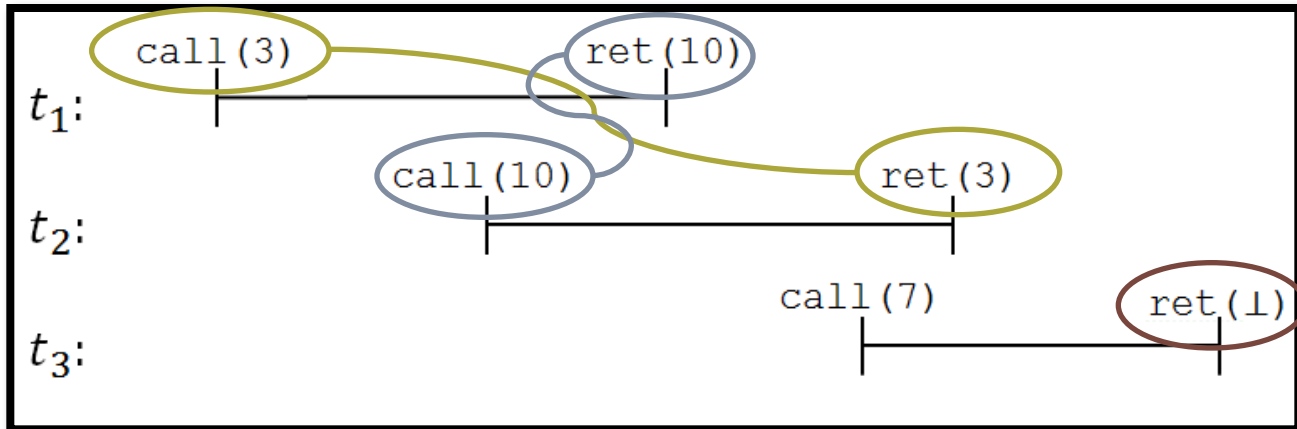
Research problem

- Linearizability uses **sequential specifications** to explain the behavior of objects
- For some objects, this cannot be done!
- Our contribution: **Concurrency-Aware linearizability**

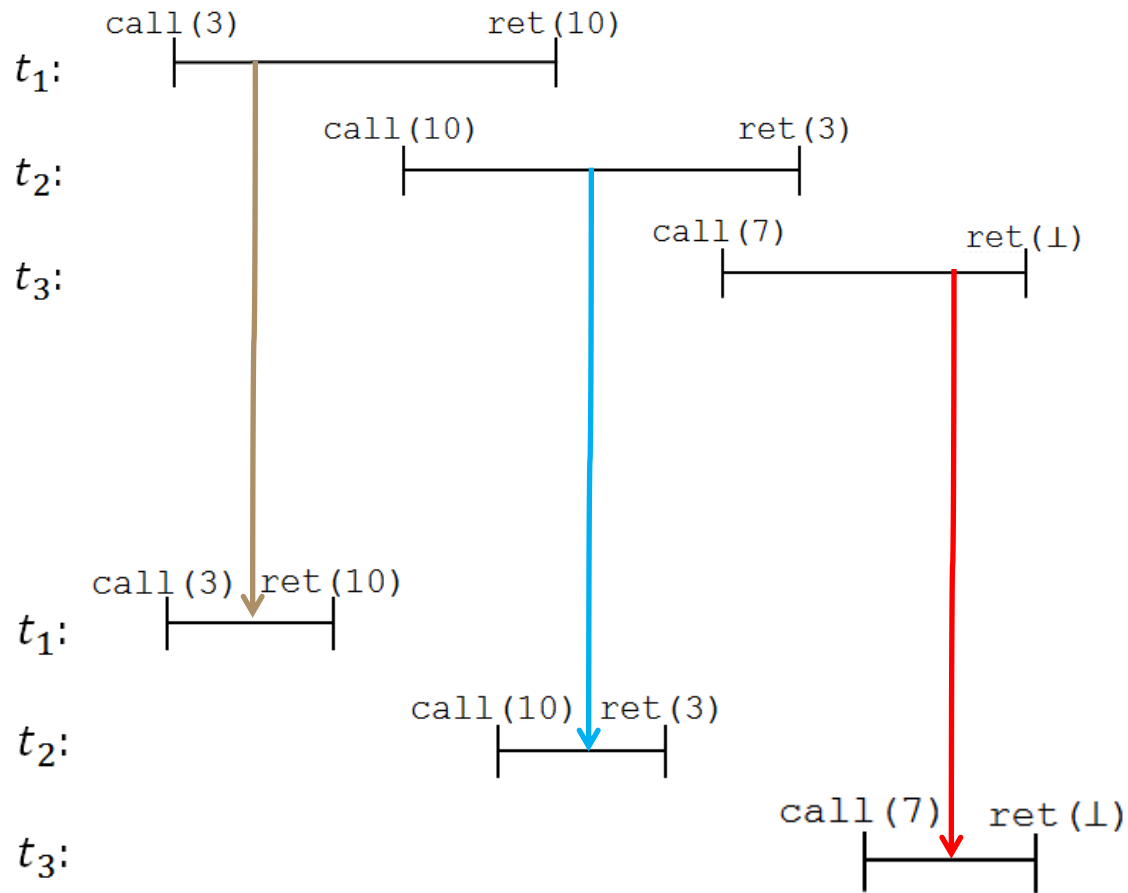
Exchanger (as in `java.util.concurrent.Exchanger`)

- Exchangers allow threads to pair up and swap elements
 - Other examples: rendezvous, elimination modules, ...

`exchange(3);` || `exchange(10);` || `exchange(7);`
 t_1 t_2 t_3



Sequential specification for Exchanger?

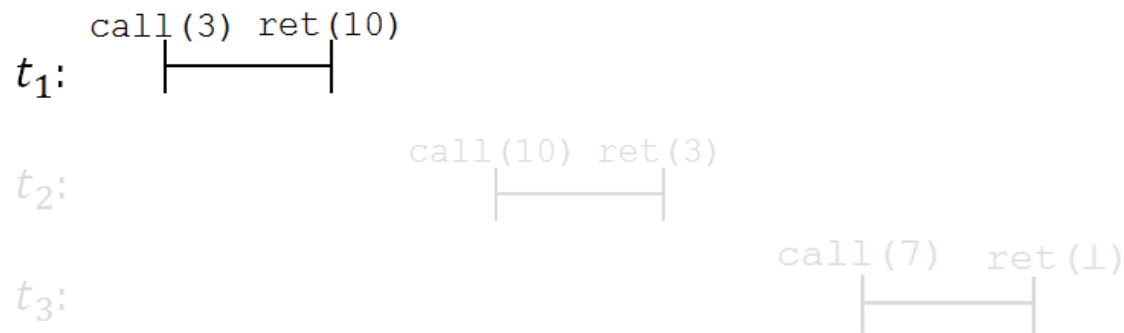


“Good specs.”: intuitive, expressive, ..., prefix-closed, ...

Sequential specification for Exchanger?

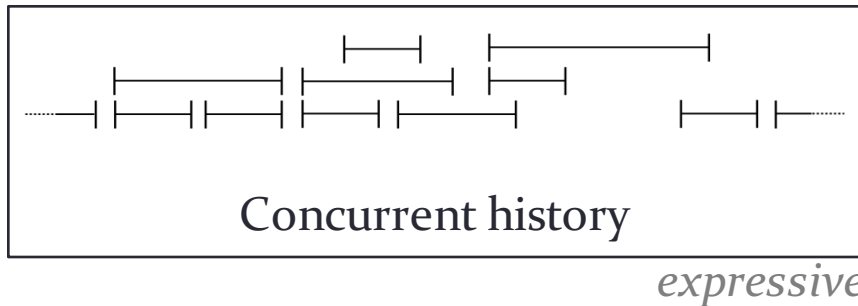
Sequential specifications for Exchanger are

- *Too lax*
- *Too strict*

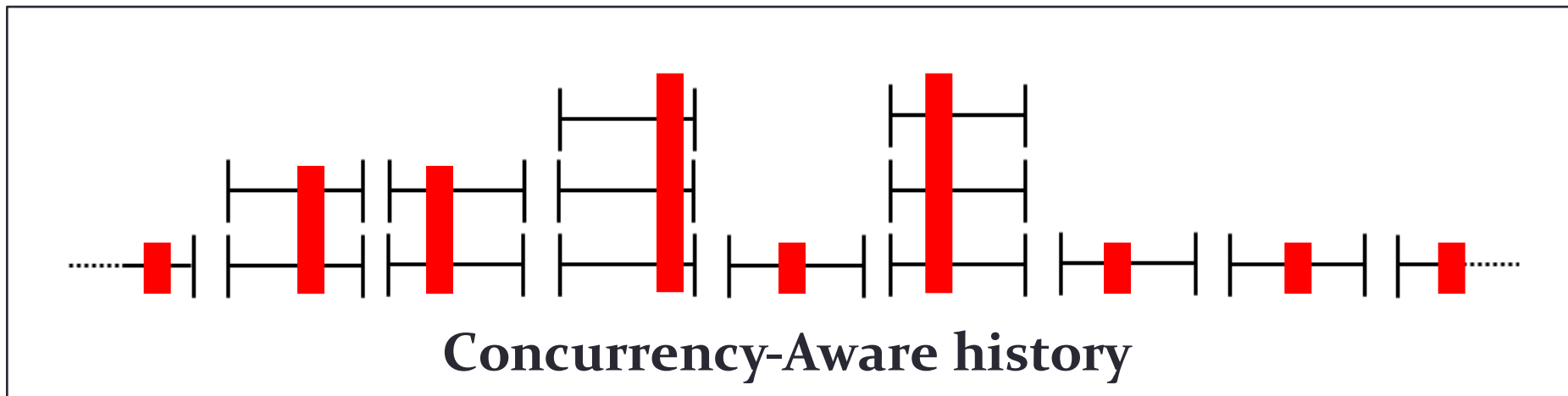
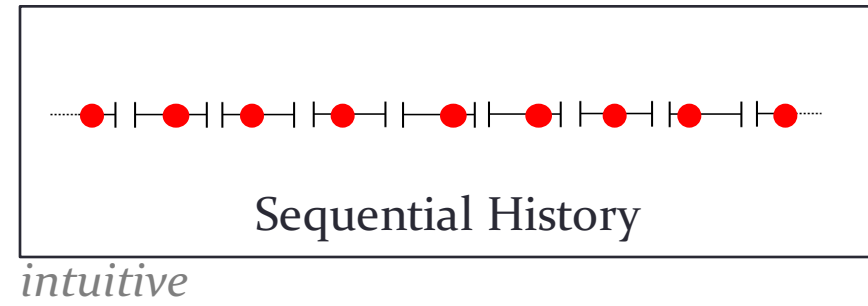


“Good specs.”: intuitive, expressive, ..., prefix-closed, ...

Concurrency-Aware Specifications

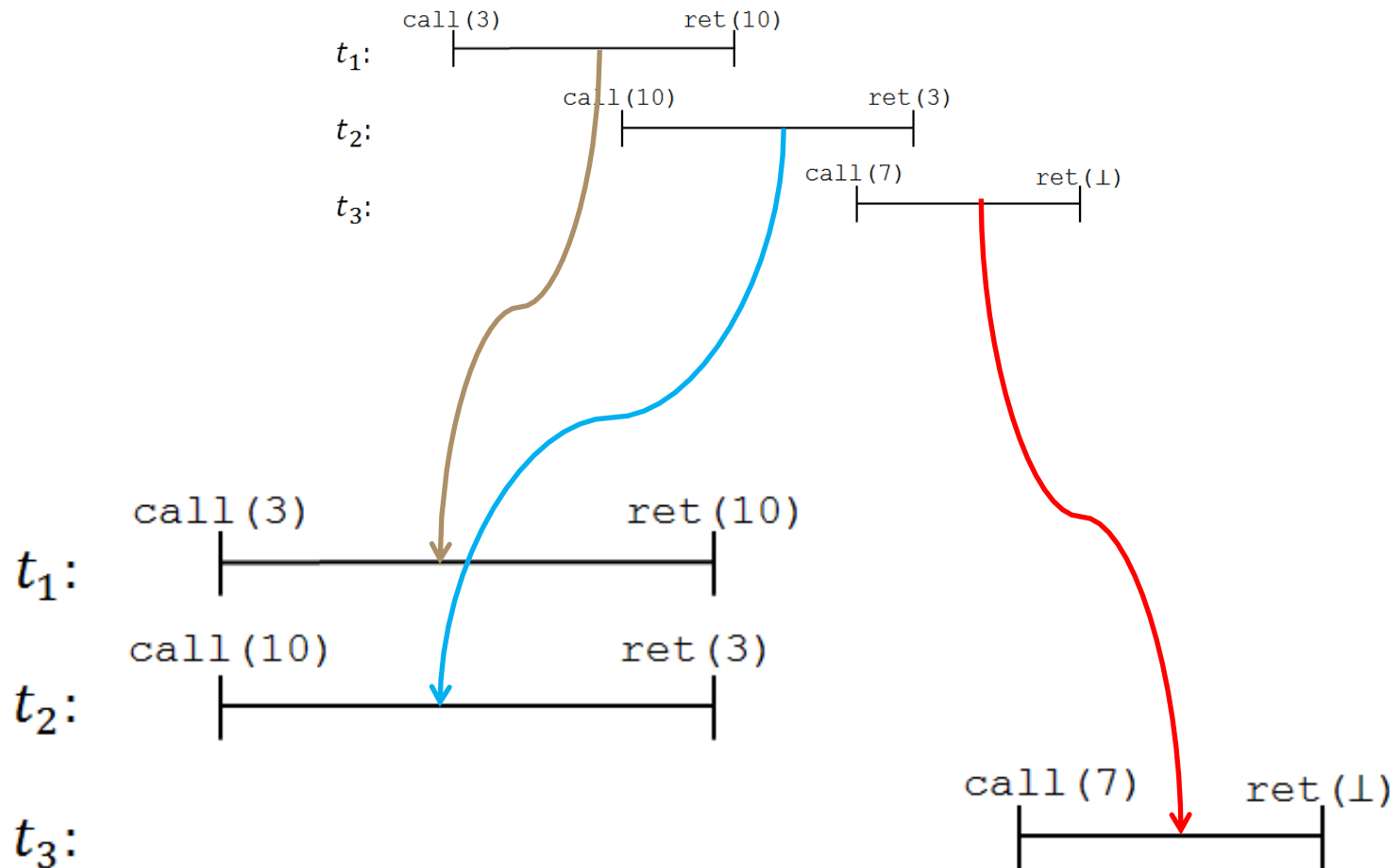


...



- **CA-specification:** a *prefix closed* set of *concurrency-aware histories*

Concurrency-Aware specification for Exchanger



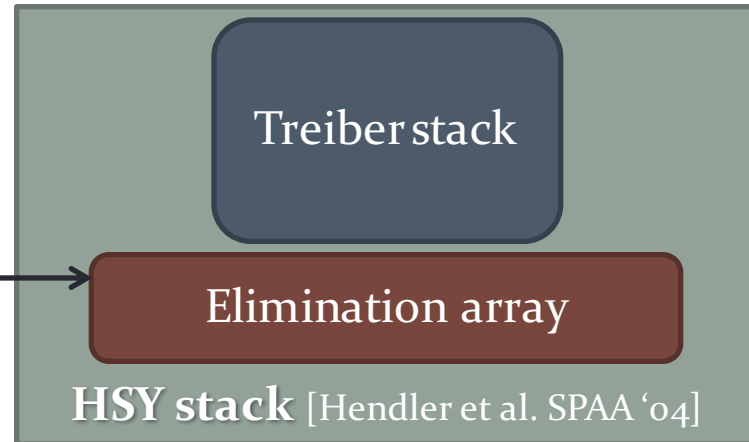
Concurrency-Aware Linearizability

- Generalizes linearizability by using **CA-histories** as the specification
- ***Concurrency-Aware Objects*** are concurrent objects that have a CA-specification

Motivation

Current work!

- Modular verification for data structures that use Concurrency-Aware objects



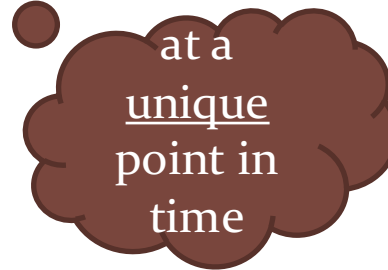
- CA-objects are used in
 - Synchronous queues [Scherer et al. PPOPP '06]
 - Scalable rendezvousing [Afek et al. DISC '11]
 - Scalable and lock-free FIFO queues [Moir et al. SPAA '05]
 - ...

Related work

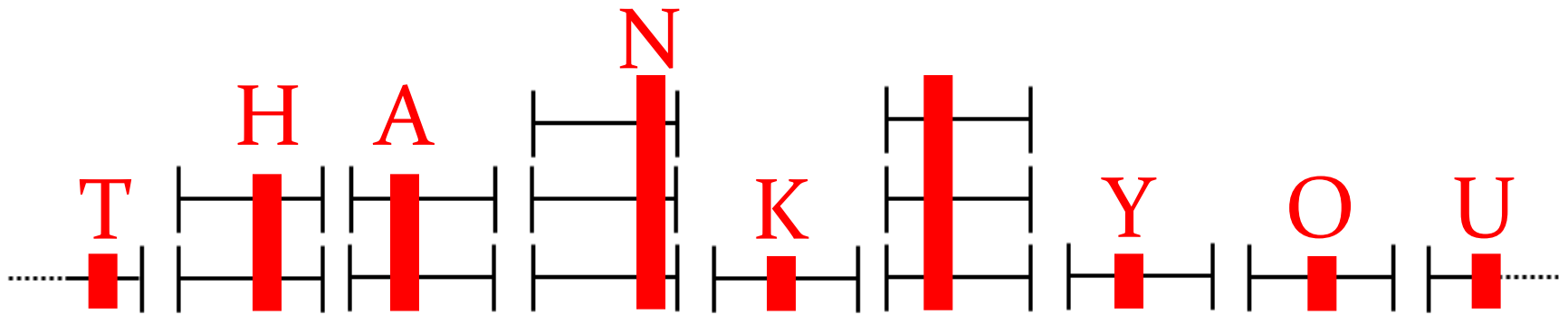
- Linearizability [Herlihy & Wing, TOPLAS '90]
- Set-Linearizability [Neiger, PODC '94]

Summary

- ***Linearizability***: “each operation appears to take effect *instantaneously* between its invocation and response”



- ***Concurrency-Aware Linearizability***: “... *possibly simultaneously*”



Concurrency-Aware Linearizability

- **CAL generalizes linearizability by using CA-histories as the specification**
 - Definition (CA-linearizability): A history H is **CA-linearizable** if there exists a **CA-history** S such that
 - $H' = \text{completionOf}(H)$: H can be completed to a history H' by removing some pending invocations or by adding responses to them
 - $H'|_t = S|_t$: In H' every thread performs the same sequence of actions as in S
 - $\prec_{\text{real-time}(H')} \subseteq \prec_{\text{real-time}(S)}$: S respects the real time order of H'
- **Concurrency-Aware Objects** are concurrent objects that have a CA-specification

The problem: sequential specifications

- Concurrent objects are considered correct if their behavior can be explained using **sequential specifications**
 - Linearizability
 - Serializability
 - Quiescent consistency
 - ...
- Simple and intuitive specifications
- Are they **expressive enough?**


Our contributions

- Identify a class of objects that do not have sequential specifications
 - *Concurrency-Aware Objects*
- Introduce ***Concurrency-Aware specifications***
 - Generalizes sequential specifications
- Present ***Concurrency-Aware Linearizability***
 - Generalizes linearizability

Summary

Linearizability:

*“each operation appears to take effect
instantaneously • • •
between its invocation and response”*



at unique
points in
time

Concurrency-Aware Linearizability:

“
...
possibly simultaneously”