Linearizability : A Correctness Condition for Concurrent Objects

Maurice P. Herlihy and Jeannette M. Wing
Carnegie Mellon University

Presented by :
Devashish Purandare
UC Santa Cruz
Oct 5th 2018

Outline

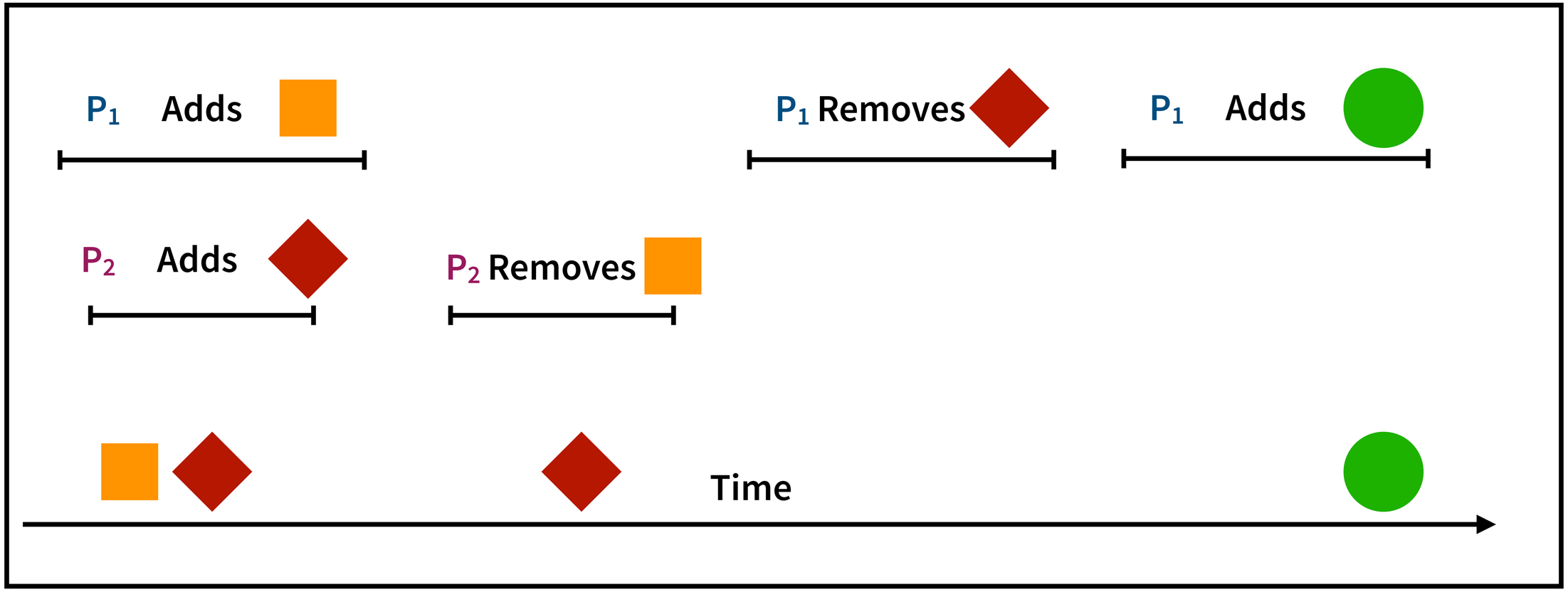
"A concurrent object is a data object shared by concurrent processes."

"Linearizability is a correctness condition for concurrent objects that exploits the semantics of abstract data types."

The FIFO Example

P_1 : Process 1
P_2 : Process 2

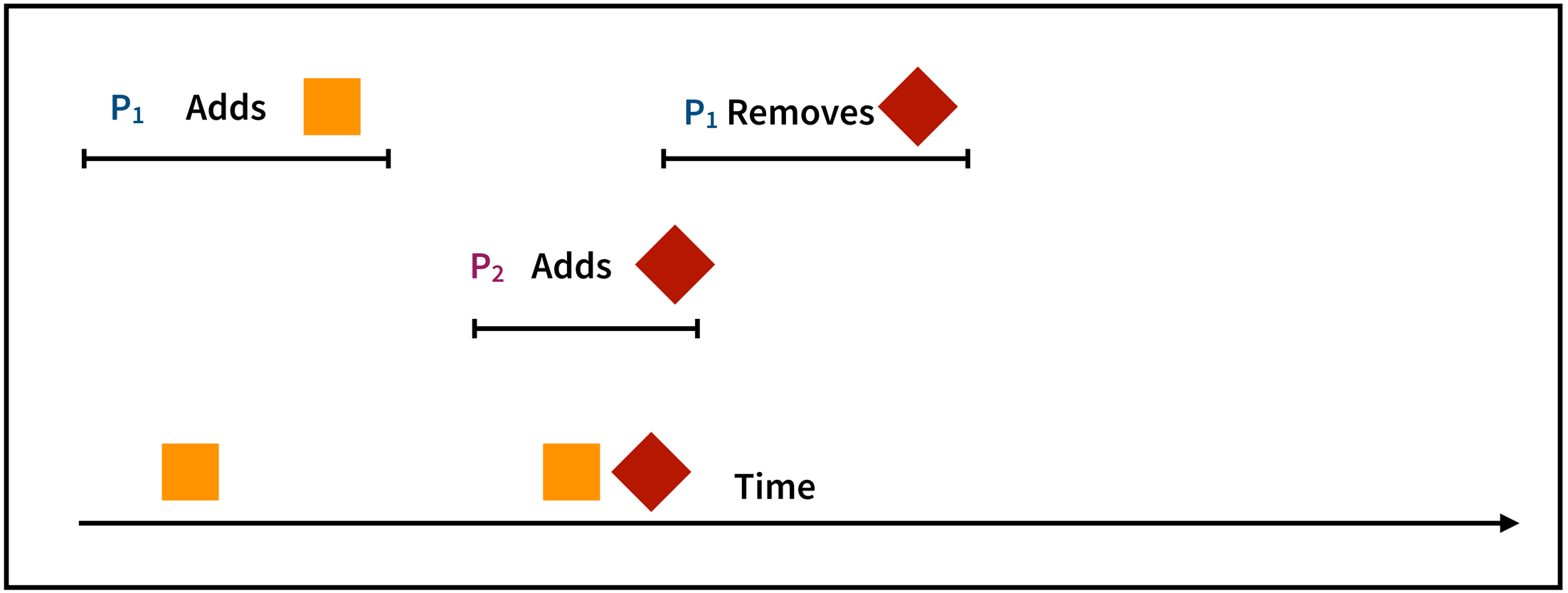
Acceptable



The FIFO Example

P_1 : Process 1
P_2 : Process 2

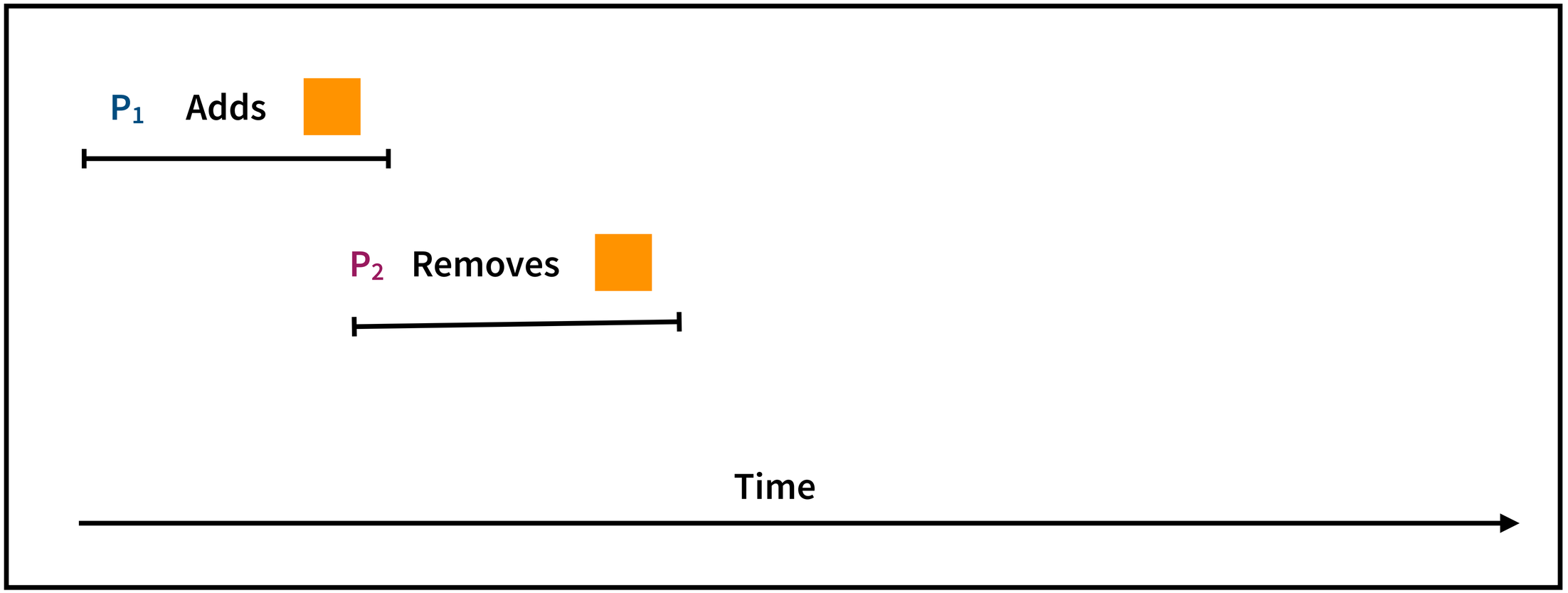
Invalid!



The FIFO Example

P_1 : Process 1
P_2 : Process 2

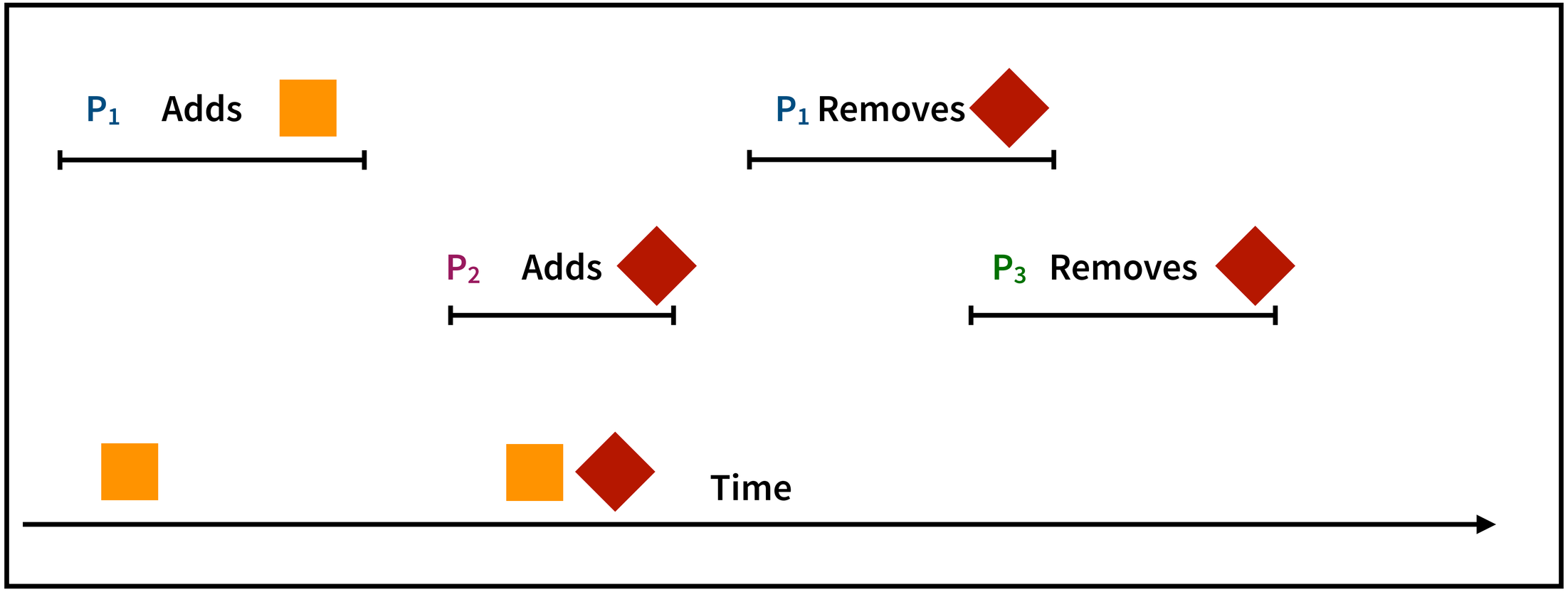
Acceptable!



The FIFO Example

P_1 : Process 1
P_2 : Process 2

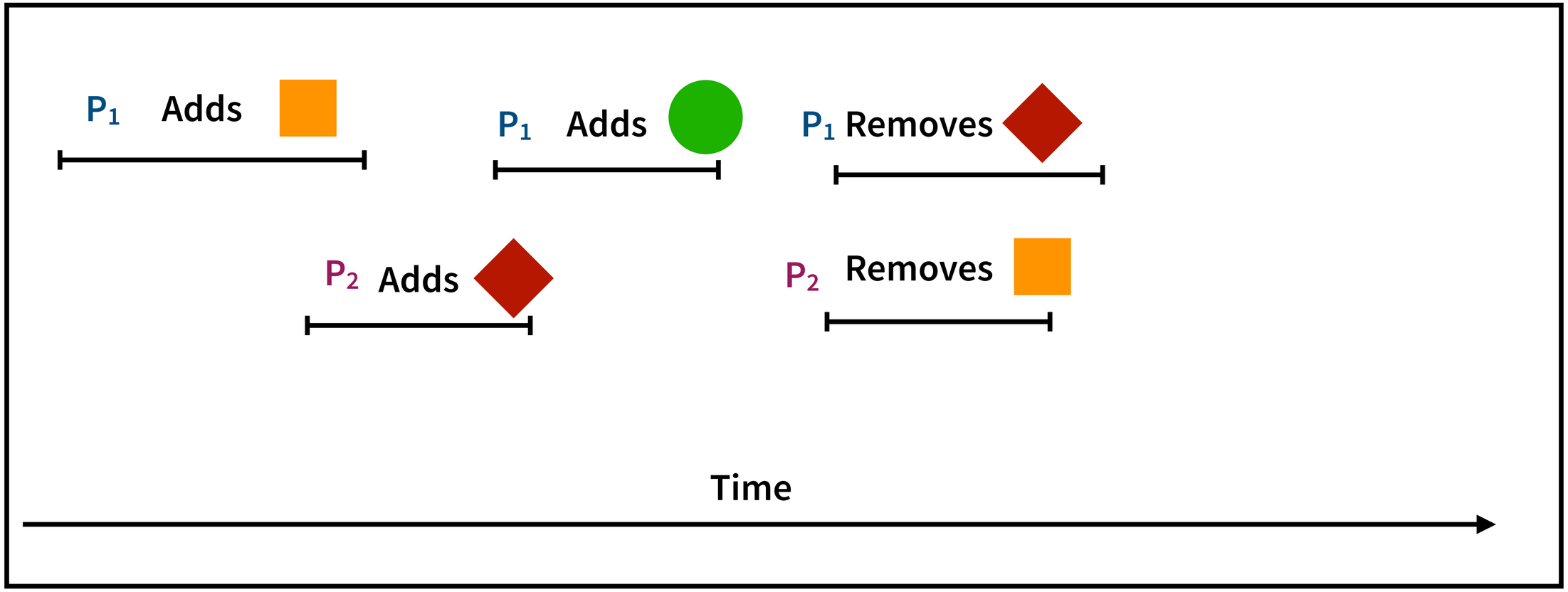
Unacceptable!



Question for the Class

P_1 : Process 1
P_2 : Process 2

Acceptable?



Formalizability of Linearizability

Invocation $inv(e)$

$$\langle x\ op(args^*)\ A \rangle$$

Response $res(e)$

$$\langle x\ term(res^*)\ A \rangle$$

Formalizability of Linearizability

$$e_0 <_H e_1$$

If $res(e_0)$ before $inv(e_1)$

Linearizability : Locality

"H is linearizable if and only if, for each object x, H | x is linearizable."

- Allows concurrent objects
- Individual modules can be separately
  - Implemented
  - Verified
  - Executed

Linearizability : Blocking

- Linearizability is nonblocking
- Totally defined operations are self contained, and don’t need to wait for other operations.
- Totally defined : invocation and response
- “Linearizability is important for concurrency and real-time response”
  - Is it?

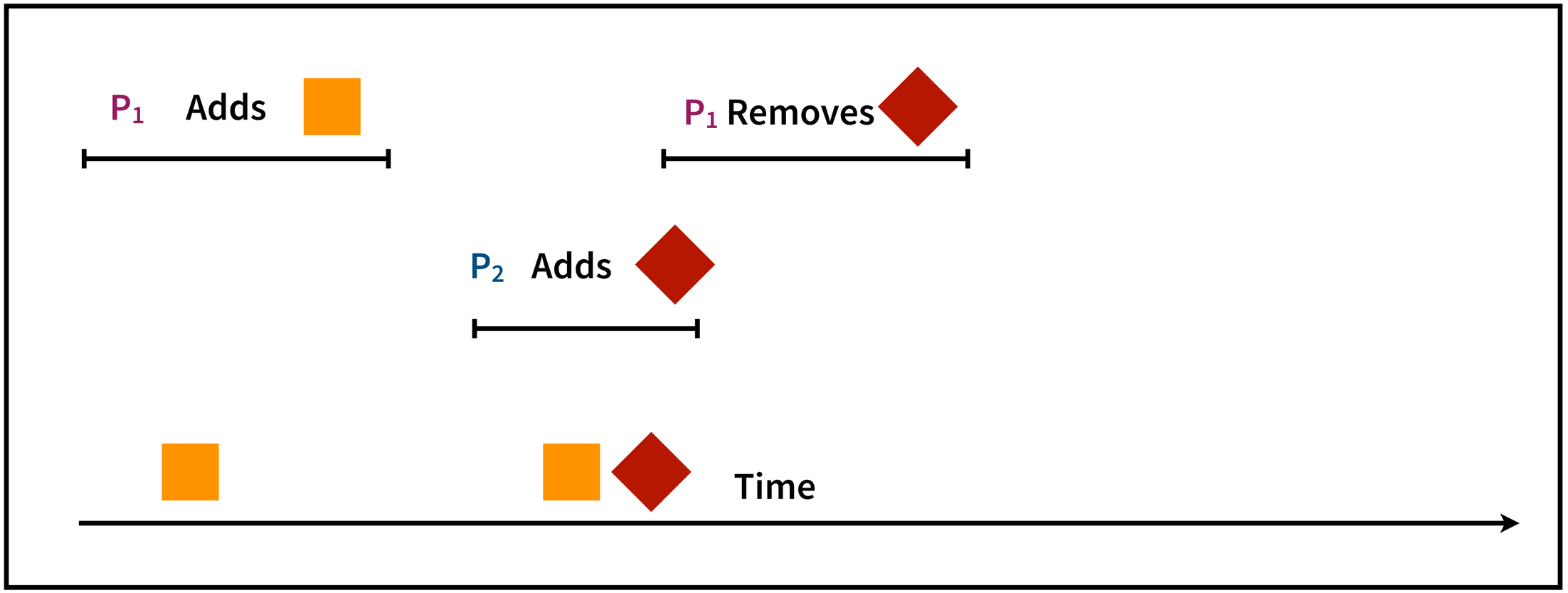
Linearizability vs Sequential Consistency

- Sequential Consistency : History is equivalent to a legal S
  - Doesn’t require ordering to be preserved

Linearizability vs Sequential Consistency

P_1 : Process 1
P_2 : Process 2

Remember?



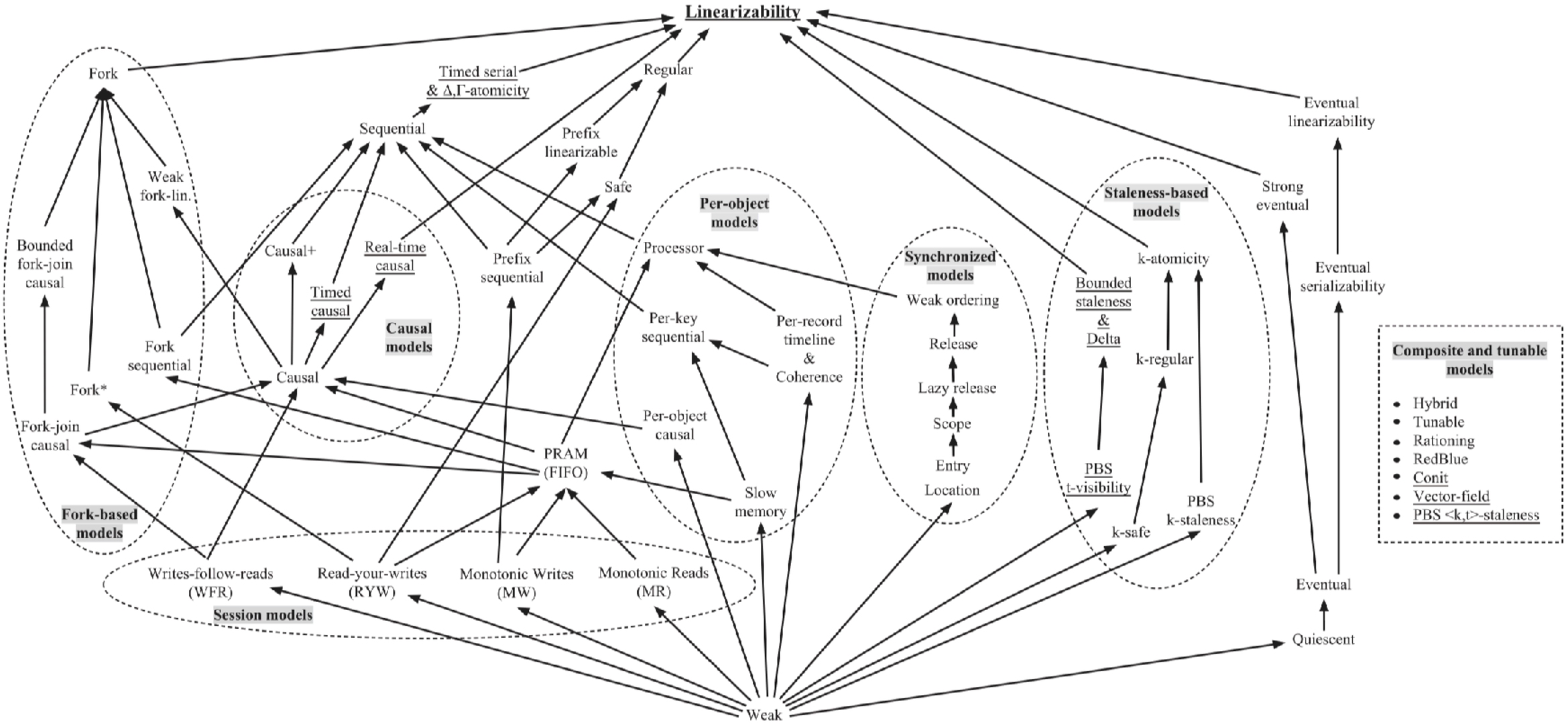
Linearizability vs Sequential Consistency

- Sequential Consistency : History is equivalent to a legal S
  - Doesn’t require ordering to be preserved
- Sequential Consistency is NOT a local property
- Sequential Consistency can be blocking

Linearizability vs Serializability

- Linearizability is a special case of strict serializability
- The ‘strict’ case being : single operation to single object
- Serializability is not strict, and it is a blocking property
- Not even strict serializability is local
- The lack of locality adds rigorous restrictions on concurrency

The Consistency Hierarchy



Strictness of Consistency

- How do we verify the implementations of linearizable objects?
- Representation Invariant (REP) and Abstraction Functions (ABS)

REP and ABS

- Representation of an object is REP with type REP
- Abstract object with type ABS are interleaved with constraints
- Invariants must be continually satisfied, and ABS should be continually defined

$H|_{ABS}$ $H|_{REP}$ Should be well formed

Methods of Verification

- We define Linearized Value as the state of the object at the end of H
- $Lin(H)$ is a set of all linearized values for H
- Proof :

$$\text{For all } r \text{ in } Lin(H|_{REP}),\ I(r) \text{ holds and } A(r) \subseteq LIN(H|_{ABS})$$

Concurrent Registers

- If r is a Read( )/Ok(u) operation in H, then there exists a Write(u)/Ok( ) operation w such that r does not precede w, and there is no other Write operation w ’ such that w precedes w ’ and w ’ precedes r.
- An interval in a history is a sequence of contiguous events. If I is an interval that does not overlap any Write operations, then all Read operations that lie within I return the same value.

Concurrent Queues

- In any sequential queue history where x is enqueued before y, x is not dequeued after y.
- If the Enq of x, Enq of y, Deq of x, and Deq of y are complete operations of H such that x's Enq precedes y's Enq, then y's Deq does not precede x's Deq (i.e., either x's Deq precedes y's, or they are concurrent).
- If the Enq of x precedes the Enq of y, and if y has been dequeued, then either x has been dequeued or there is a pending Deq concurrent with the Deq of Y.
- If x has been dequeued, then it was enqueued, and the Deq operation does not precede the Enq.

Class Questions and Conclusions

- Linearizability is a safety property
- Can help us specify, implement and verify concurrent data objects
- Where is this useful?
- Where would you not use linearizability?