COMP 322: Fundamentals of Parallel Programming

Lecture 29: Linearizability

Mack Joyner
mjoyner@rice.edu

http://comp322.rice.edu

Linearizability: Correctness of Concurrent Objects

- A *concurrent object* is an *object* that can correctly handle *methods* invoked *concurrently* by different tasks or threads
  - —e.g., AtomicInteger, ConcurrentHashMap, ConcurrentLinkedQueue, …

- For the discussion of linearizability, we will assume that the body of each method in a concurrent object is itself sequential
  - —Assume that methods do not create threads or async tasks

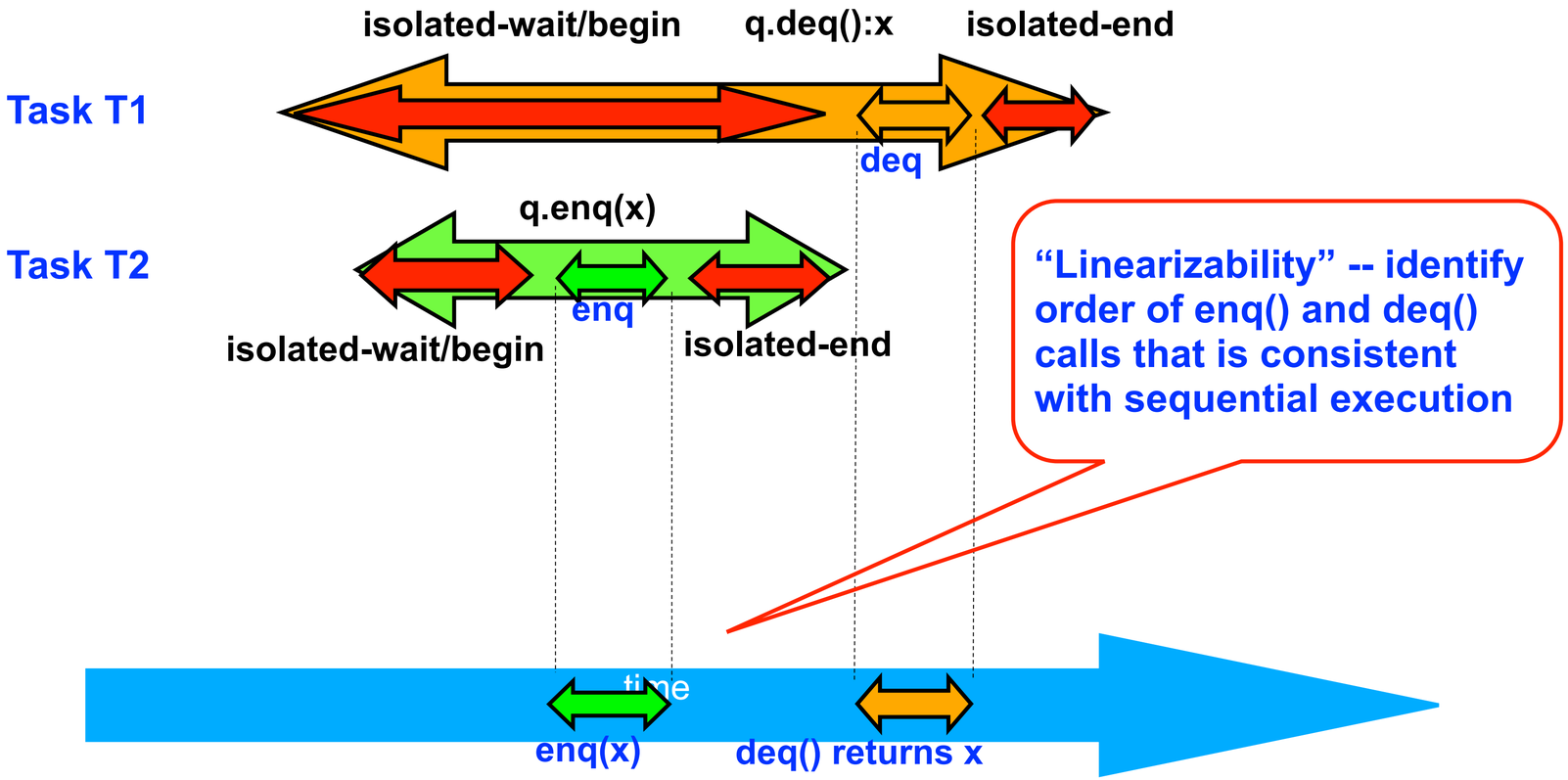
Linearizability: Correctness of Concurrent Objects

- Consider a simple FIFO (First In, First Out) queue as a canonical example of a concurrent object
  - —Method q.enq(o) inserts object o at the tail of the queue
    - Assume that there is unbounded space available for all enq() operations to succeed
  - —Method q.deq() removes and returns the item at the head of the queue.
    - Throws EmptyException if the queue is empty.

- Without seeing the implementation of the FIFO queue, we can tell if an execution of calls to enq() and deq() is correct or not, in a sequential program

- *How can we tell if the execution is correct for a parallel program?*

Linearization: Identifying a sequential order of concurrent method calls



Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

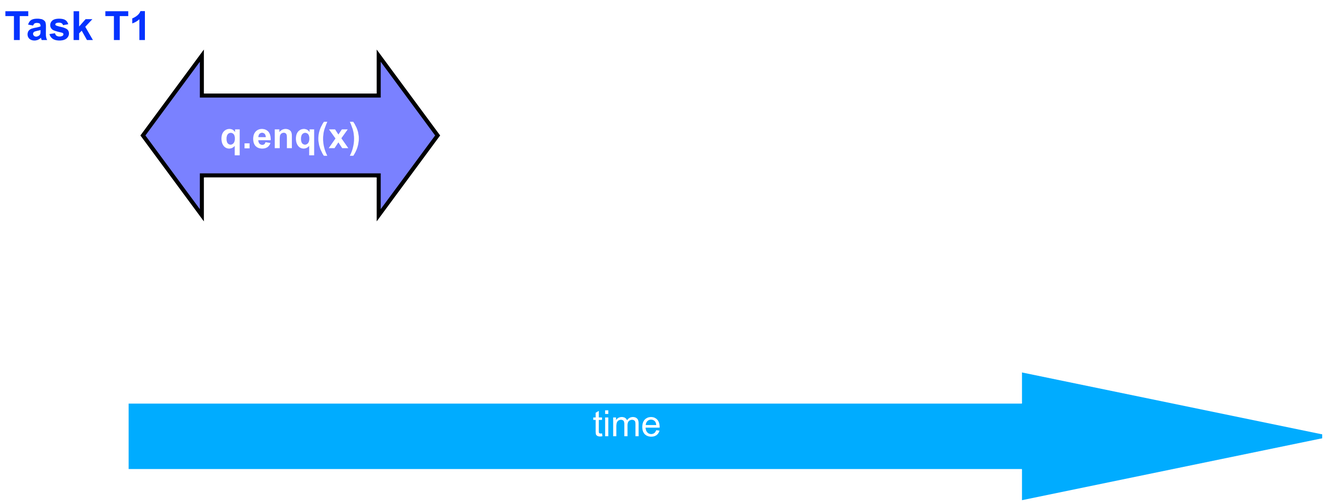
Informal Definition of Linearizability

- Assume that each method call takes effect "instantaneously" at some point in time between its invocation and return.

- An *execution (schedule) is linearizable* if we can choose *one set of* instantaneous points that is consistent with a sequential execution in which methods are executed at those points
  - It's okay if some other set of instantaneous points is not linearizable

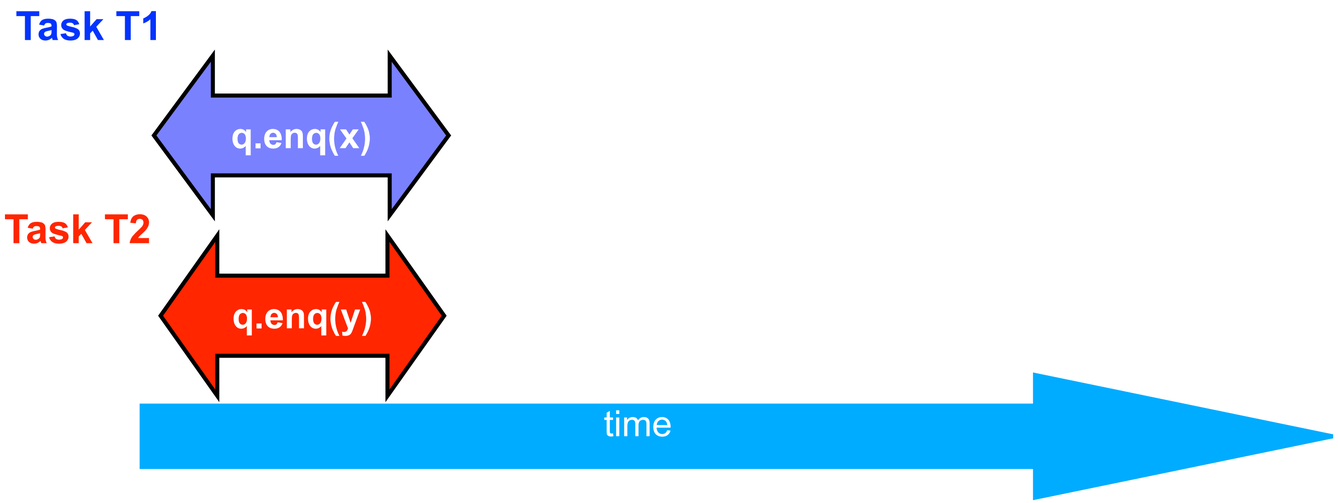
- A *concurrent object is linearizable* if all its executions are linearizable
  - Linearizability is a "black box" test based on the object's behavior, not its internals

Example 1



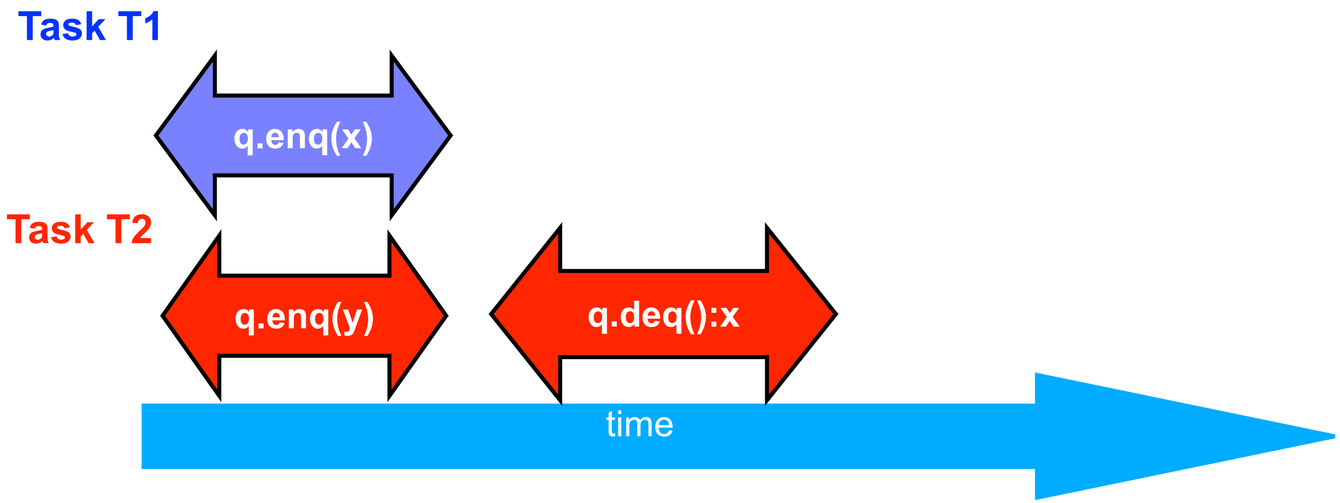
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 cont.



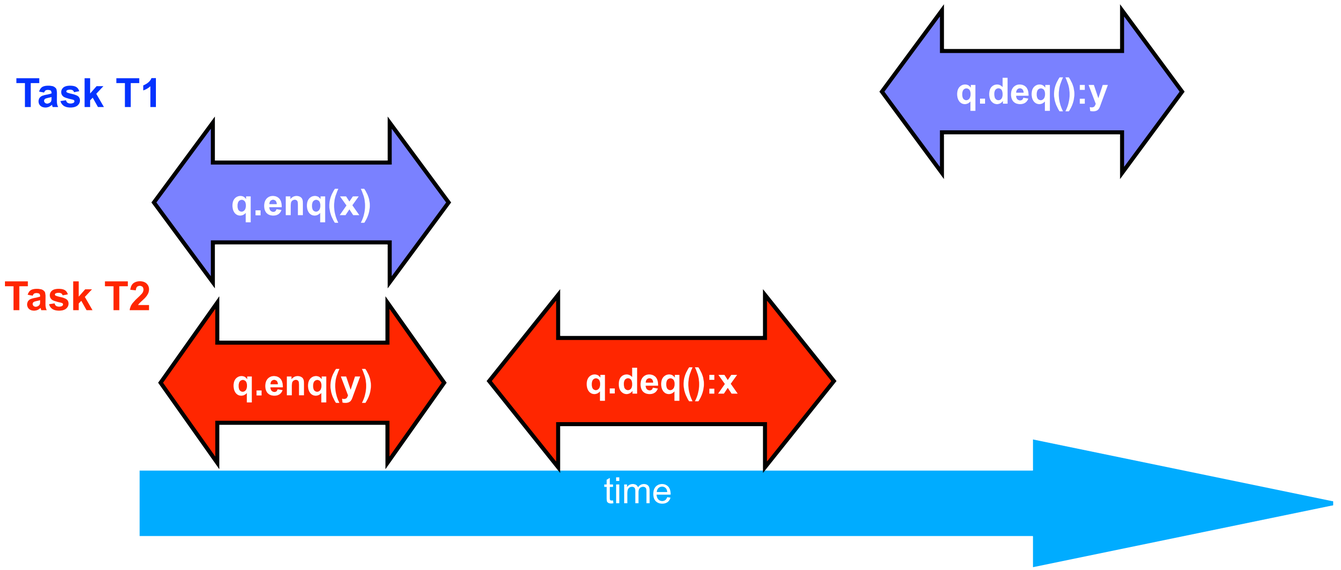
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 cont.



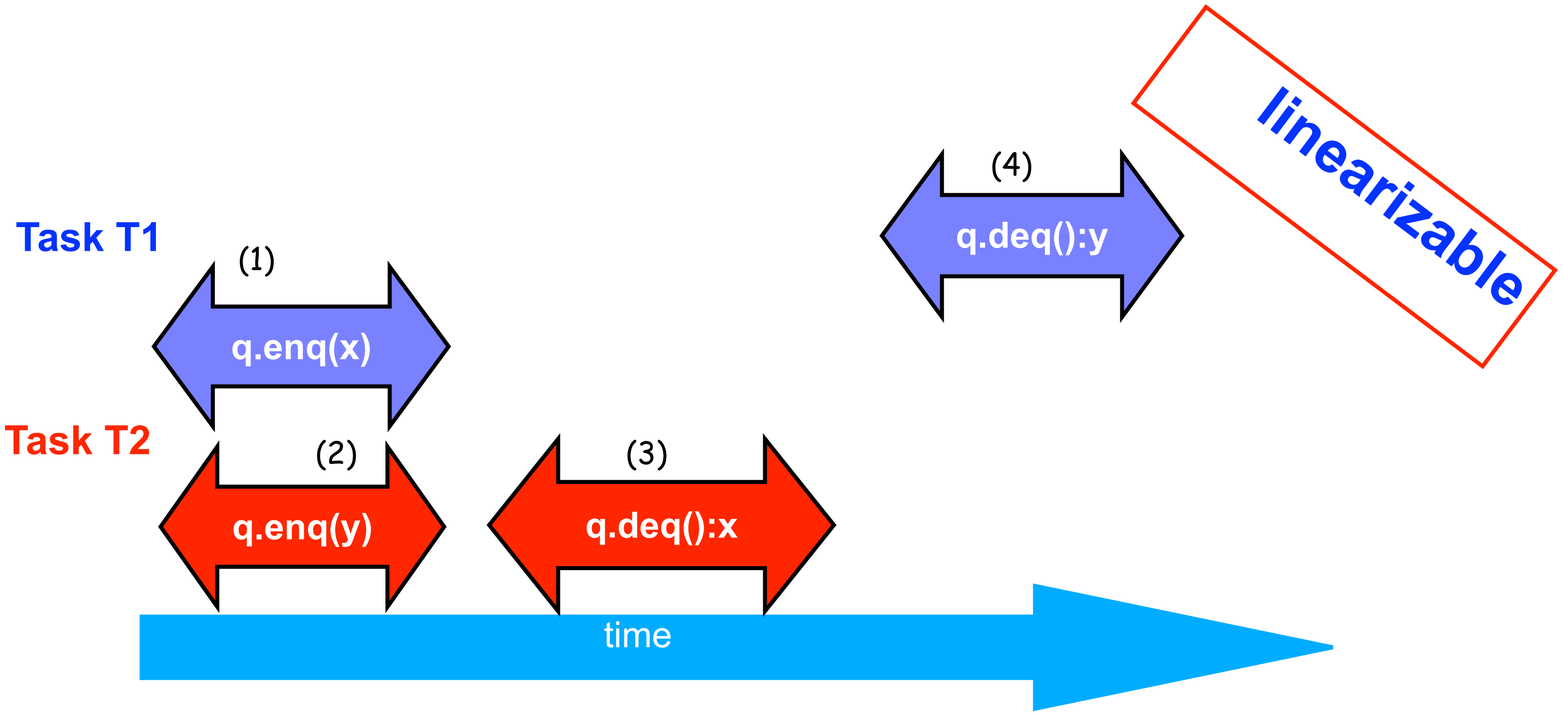
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 cont.



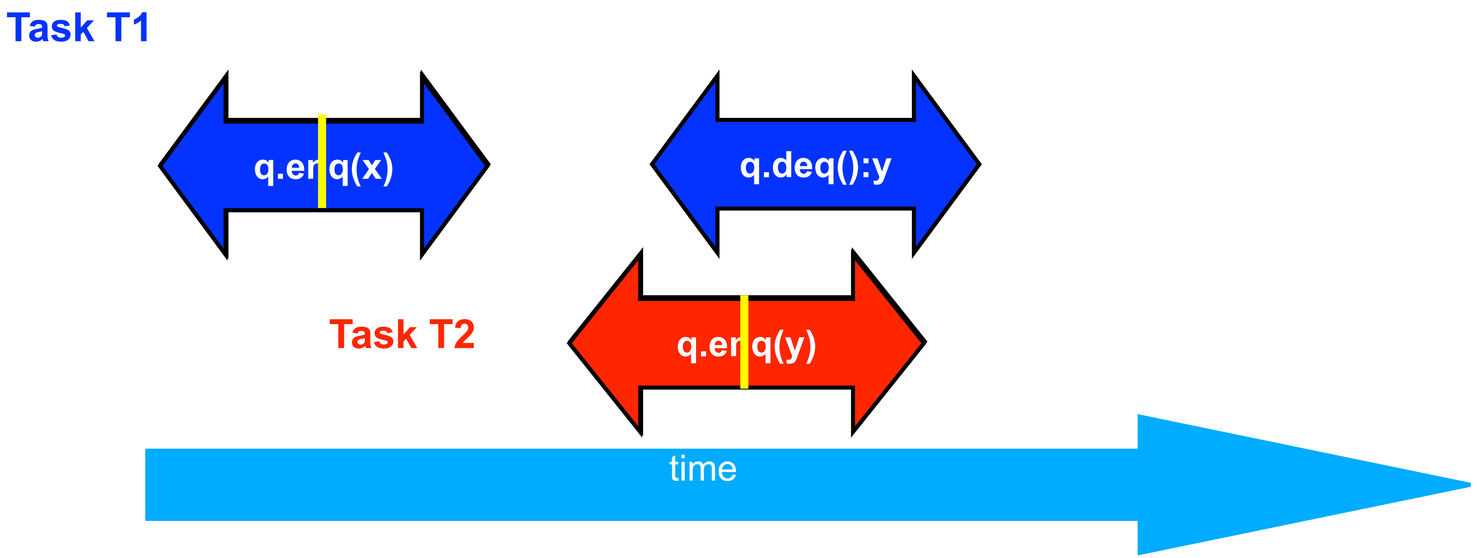
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1: is this execution linearizable?



Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

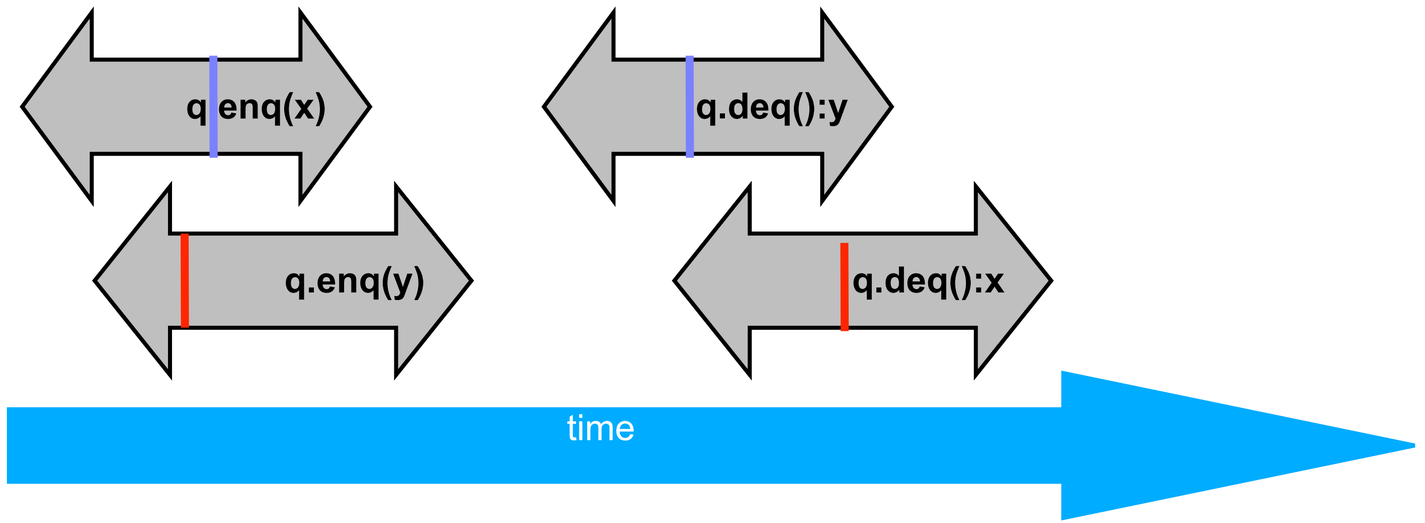
Example 2: is this execution linearizable?



Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 3

Is this execution linearizable? How many possible linearizations does it have?



Example 4: execution of isolated implementation of FIFO queue q

Is this a linearizable execution?

Time	Task A	Task B
0	Invoke `q.enq(x)`	
1	Work on `q.enq(x)`	
2	Work on `q.enq(x)`	
3	Return from `q.enq(x)`	
4		Invoke `q.enq(y)`
5		Work on `q.enq(y)`
6		Work on `q.enq(y)`
7		Return from `q.enq(y)`
8		Invoke `q.deq()`
9		Return `x` from `q.deq()`

Linearizability of Concurrent Objects (Summary)

Concurrent object

- A concurrent object is an object that can correctly handle methods invoked in parallel by different tasks or threads
  - —Examples: Concurrent Queue, AtomicInteger

Linearizability

- Assume that each method call takes effect “instantaneously” at some distinct point in time between its invocation and return.
- An execution is linearizable if we can choose instantaneous points that are consistent with a sequential execution in which methods are executed at those points
- An object is linearizable if all its possible executions are linearizable