COMP 322: Fundamentals of Parallel Programming

Lecture 25: Concurrent Objects and the Linearizability Property

Vivek Sarkar, Shams Imam
Department of Computer Science, Rice University

Contact email: vsarkar@rice.edu, shams.imam@twosigma.com

http://comp322.rice.edu/

Worksheet #24 solution: Bounded Buffer Example

Consider the case when multiple threads call insert() and remove() methods concurrently for a single BoundedBuffer instance with SIZE >= 1.

1) Can you provide an example in which the wait set includes a thread waiting at line 2 in insert() and a thread waiting at line 11 in remove(), in slide 11? If not, why not?

No, only producer threads enter the wait set when the buffer is full, and only consumer threads enter the wait set when the buffer is empty

2) How would the code behave if all wait/notify calls (lines 2, 6, 11, 15) were removed from the insert() and remove() methods in slide 11?

insert() may overwrite existing elements when buffer is supposed to be full

remove() may return undefined values when buffer is supposed to be empty

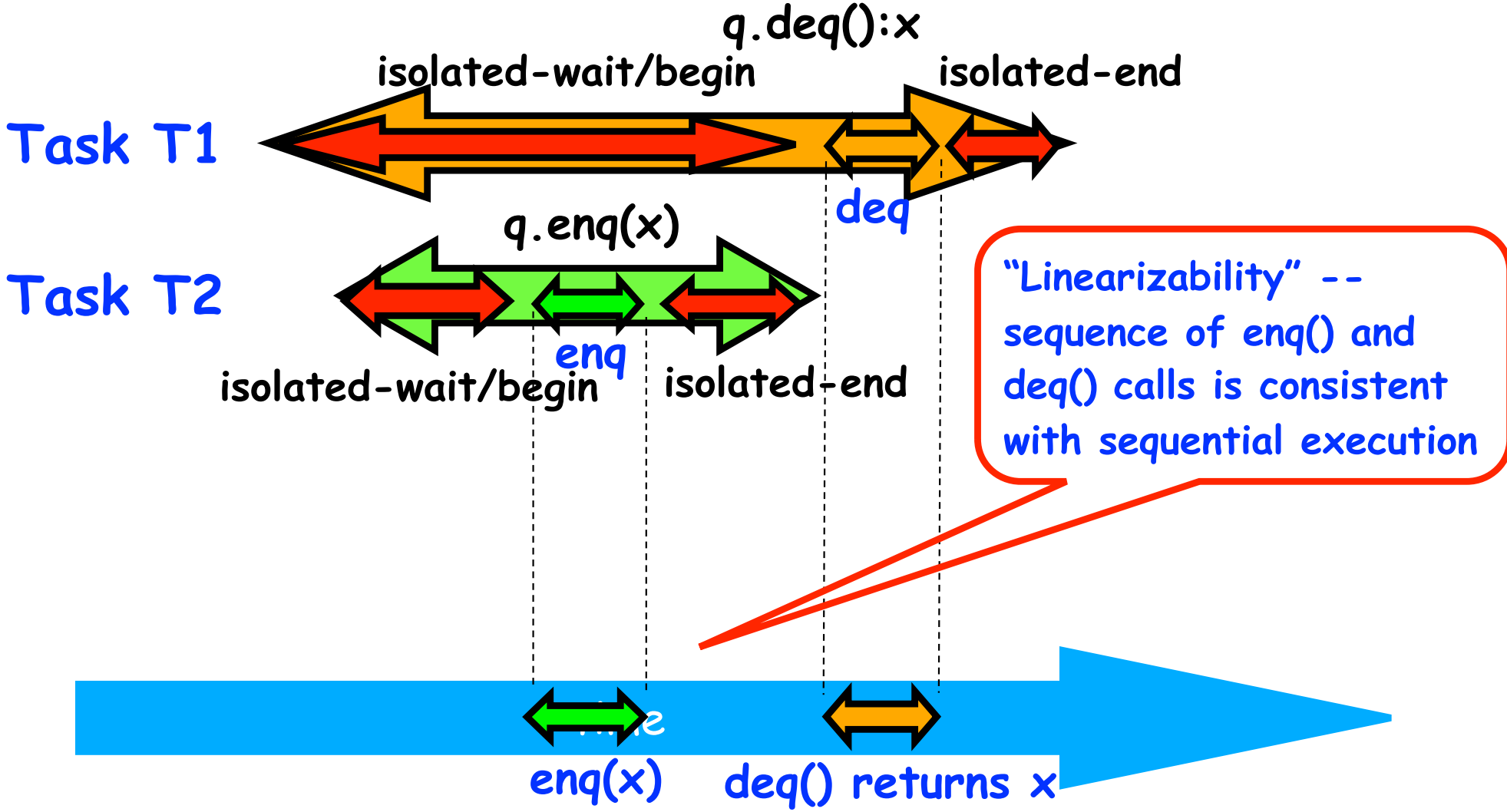
Concurrent Objects

- A *concurrent object* is an *object* that can correctly handle *methods* invoked *concurrently* by different tasks or threads
  - Also referred to as "thread-safe objects"
  - e.g., AtomicInteger, ConcurrentHashMap, BoundedBuffer, …
- 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 child async tasks
- Implementations of methods can be serialized (e.g., using synchronized or object-based isolated statements) or can be concurrent (e.g., by using read-write modes in object-based isolation)
- A desirable goal is to develop implementations that are concurrent for performance while being as close to the semantics of the serial version as possible

Correctness of a Concurrent Object

- 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.
- What does it mean for a concurrent object like a FIFO queue to be correct?
  - What is a concurrent FIFO queue?
  - FIFO implies a strict temporal order
  - Concurrent implies an ambiguous temporal order

Linearization: identifying a sequential order of concurrent method calls



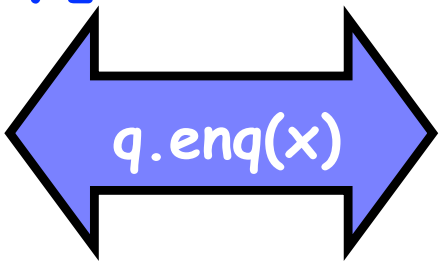
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

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

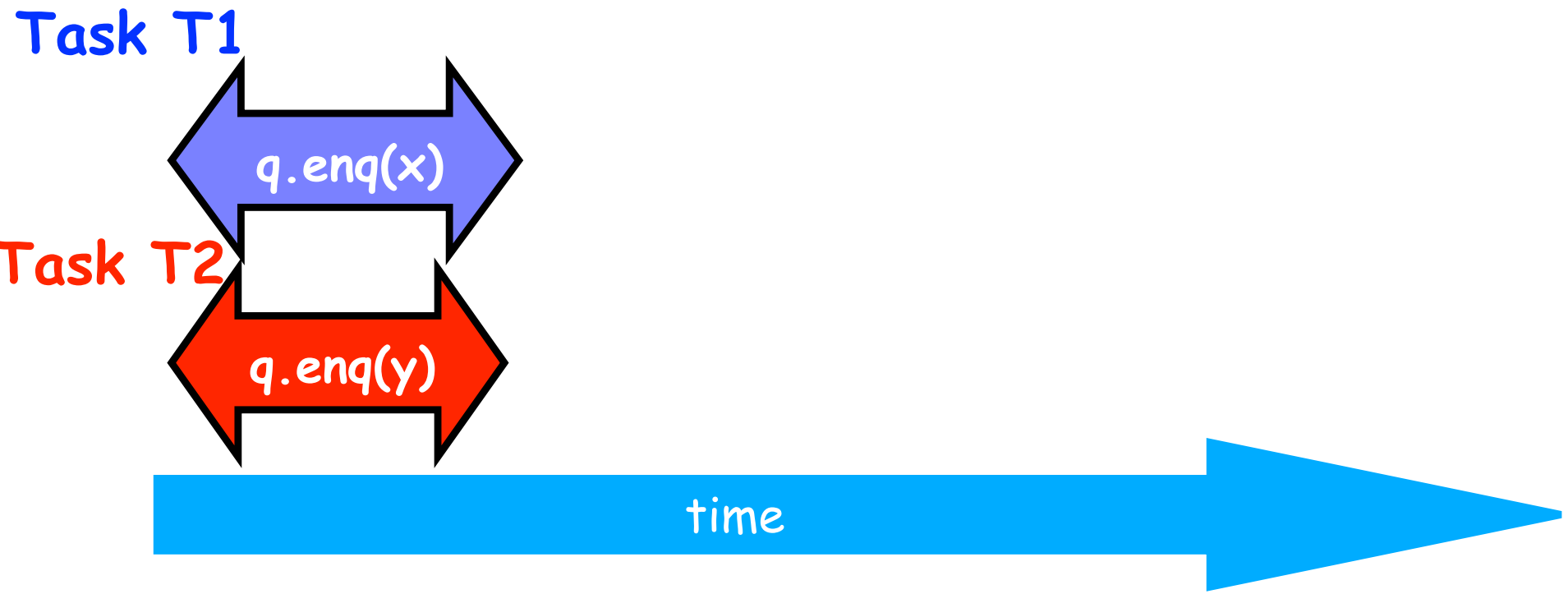
Example 1

Task T1



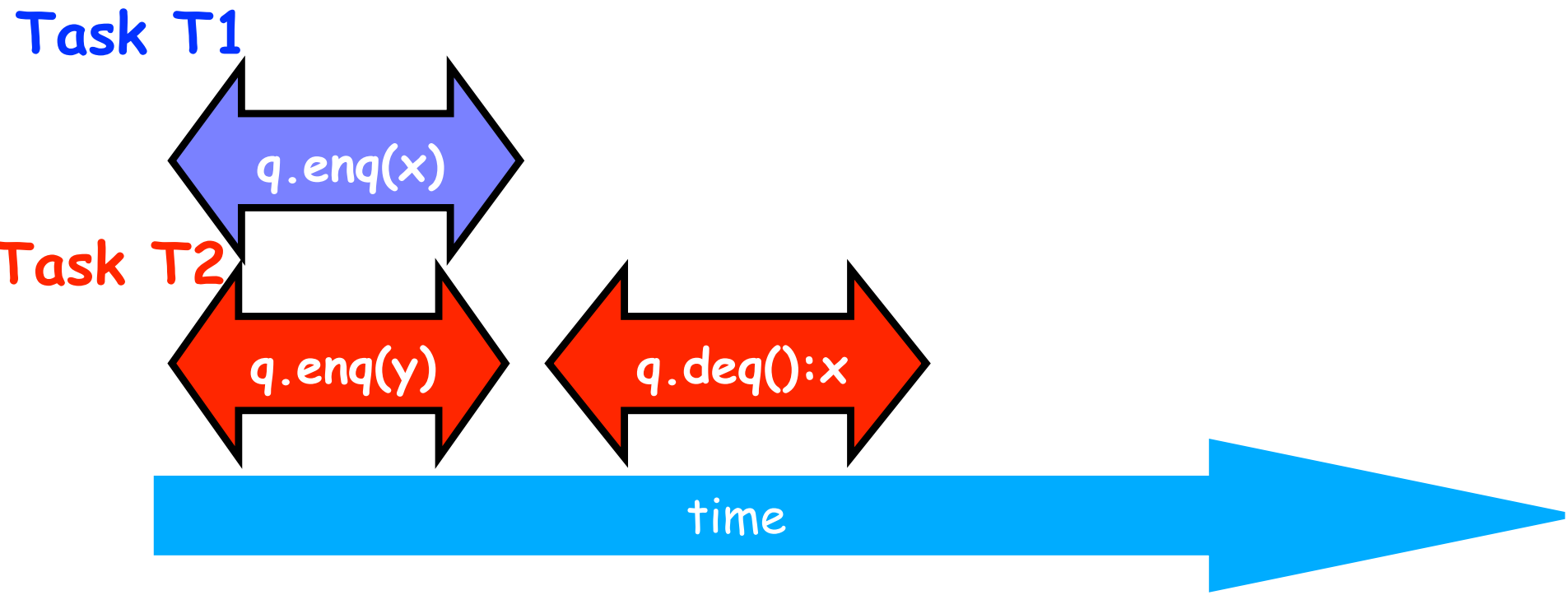
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 (contd)



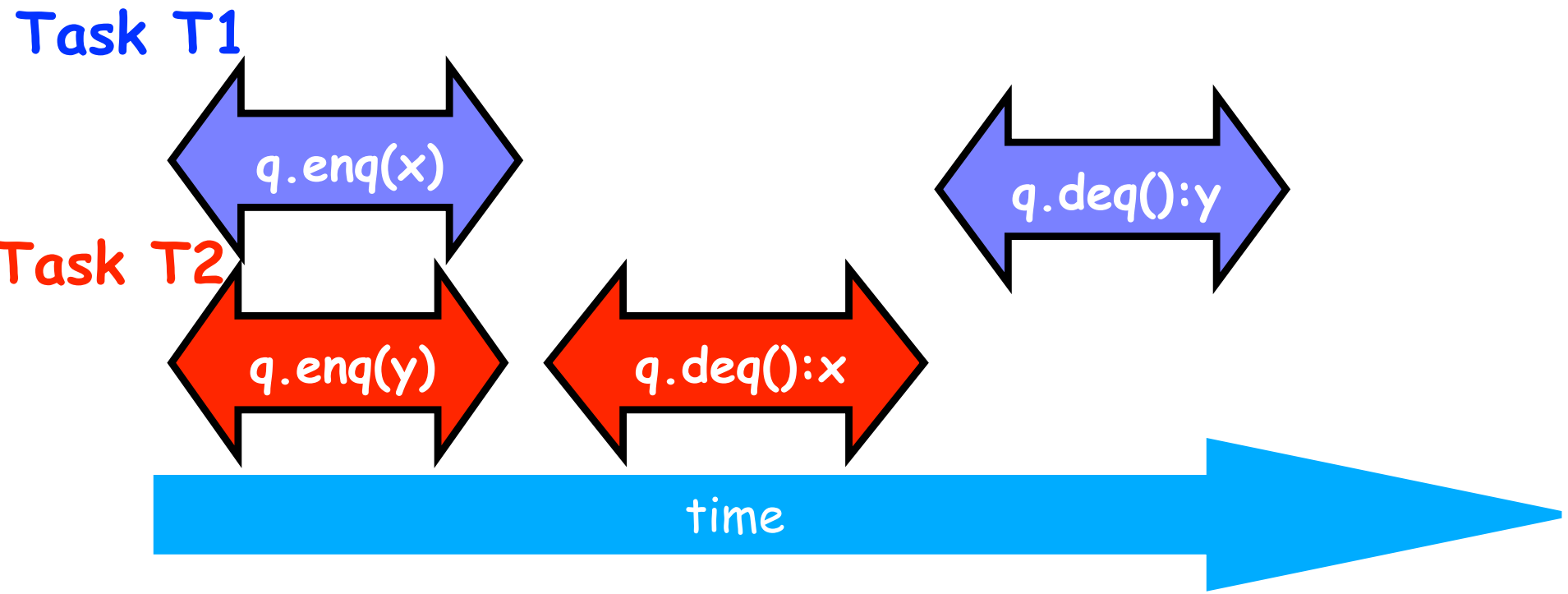
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 (contd)



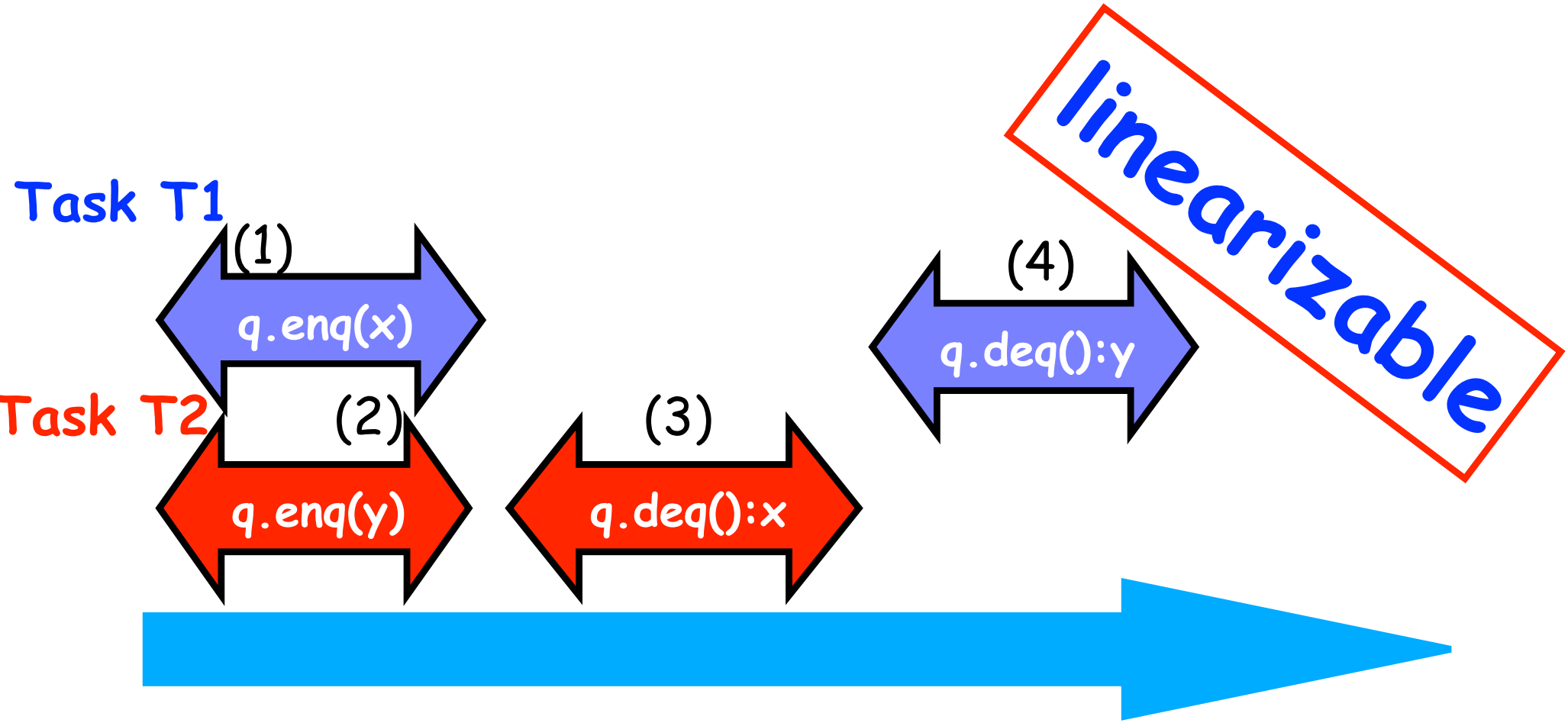
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 (contd)



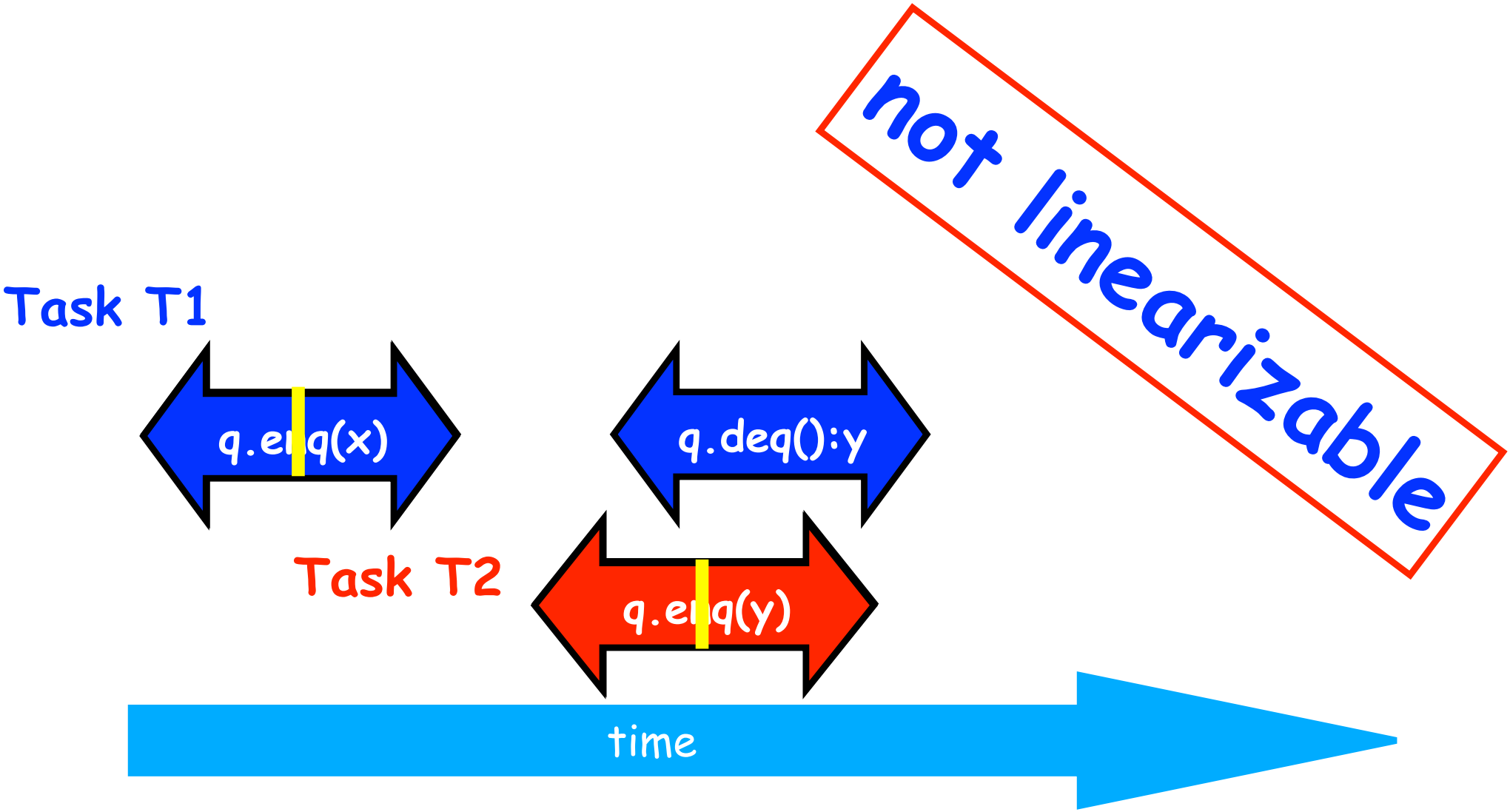
Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt

Example 1 (contd)



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

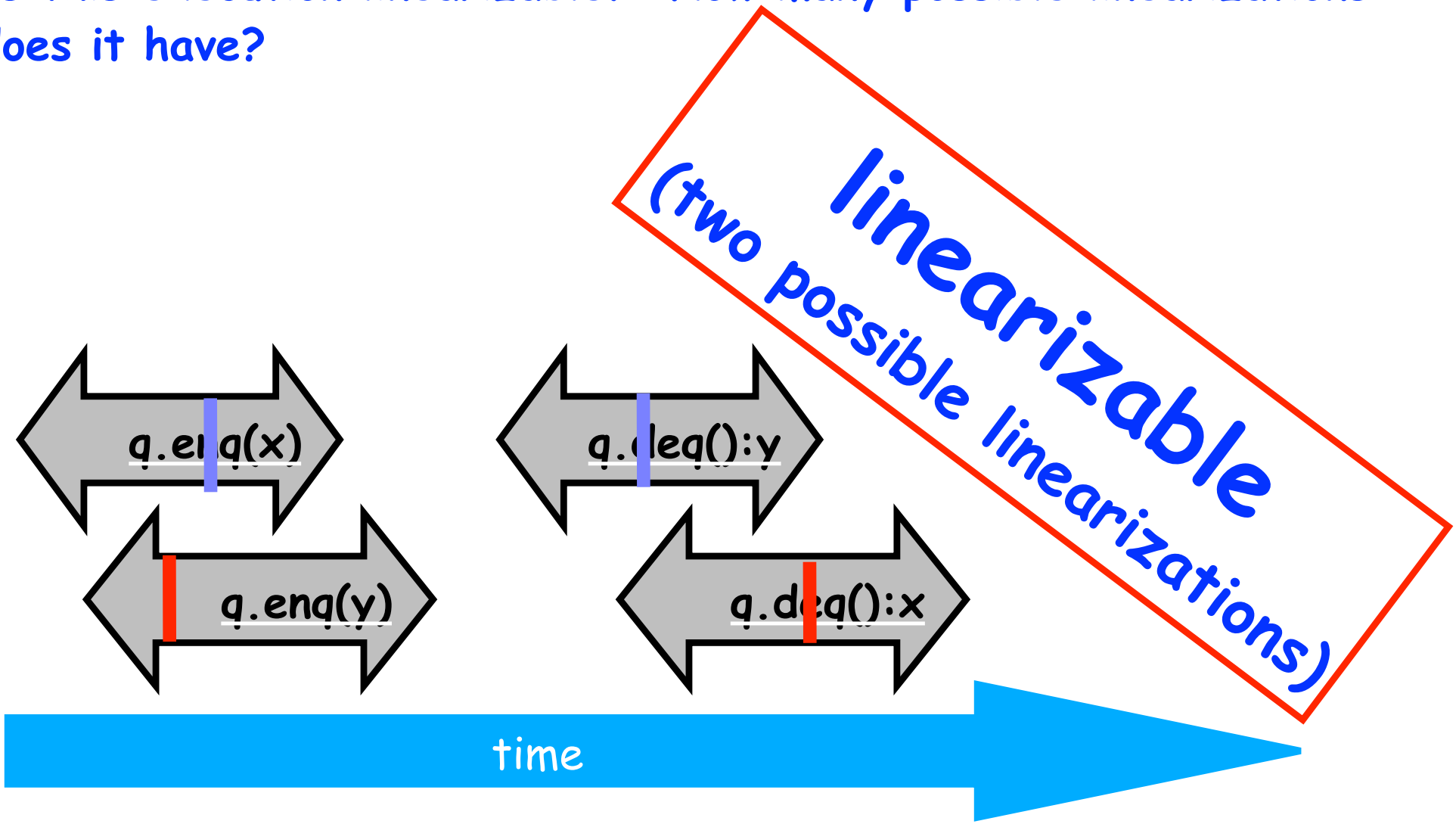
Example 2: is this execution linearizable?



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 an 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()`

Yes! Can be linearized as "q.enq(x) ; q.enq(y) ; q.deq():x".

Example 5: execution of a concurrent implementation of a FIFO queue q

Is this a linearizable execution?

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

Yes! Can be linearized as "q.enq(x) ; q.enq(y) ; q.deq():x".

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