Lecture 12

- Administration

Correctness of Concurrent Objects

- Quiescent Consistency
- Sequential Consistency
- Linearizability

Principles

- Principle 1

  Method call should appear to happen in a one-at-a-time sequential order

- Principle 2

  Method calls separated by a period of quiescence should appear to take effect in real-time order.

- Principle 3

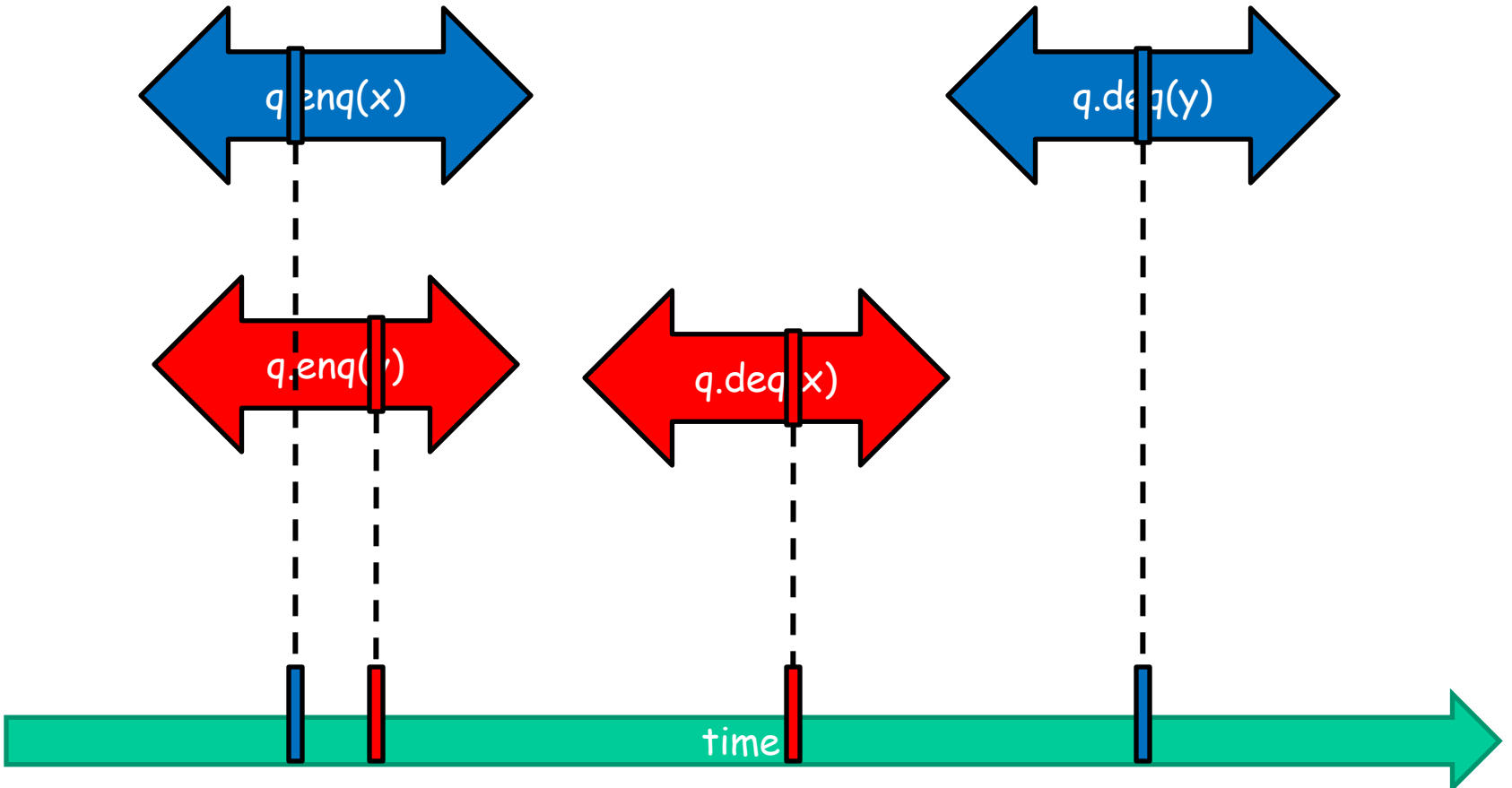
  Method calls should appear to take effect in program order

- Principle 4

  Each method call should appear to be instantaneous at some moment between its invocation and response.

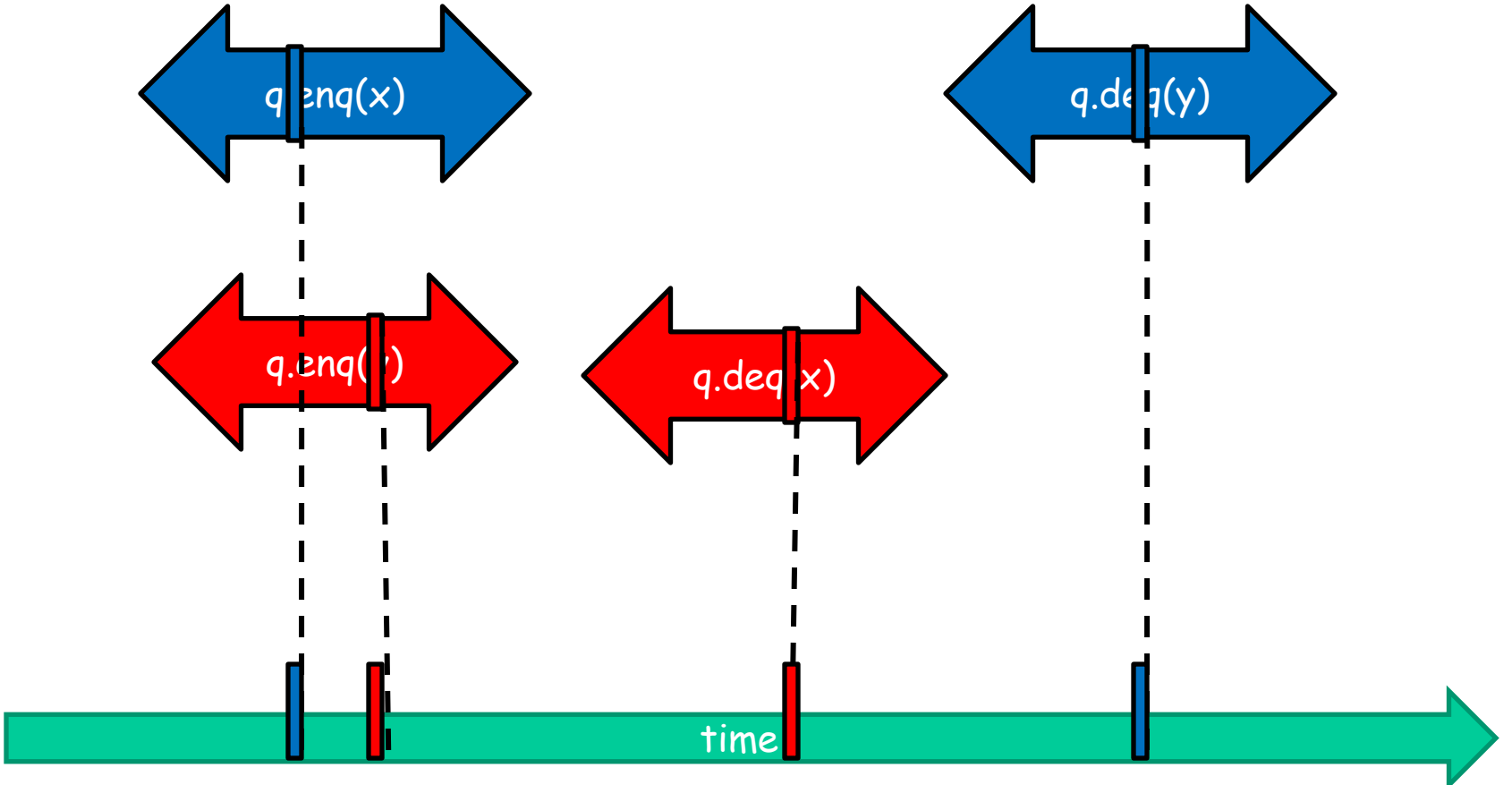
Linearizablity

Is this linearizable? **Yes!**



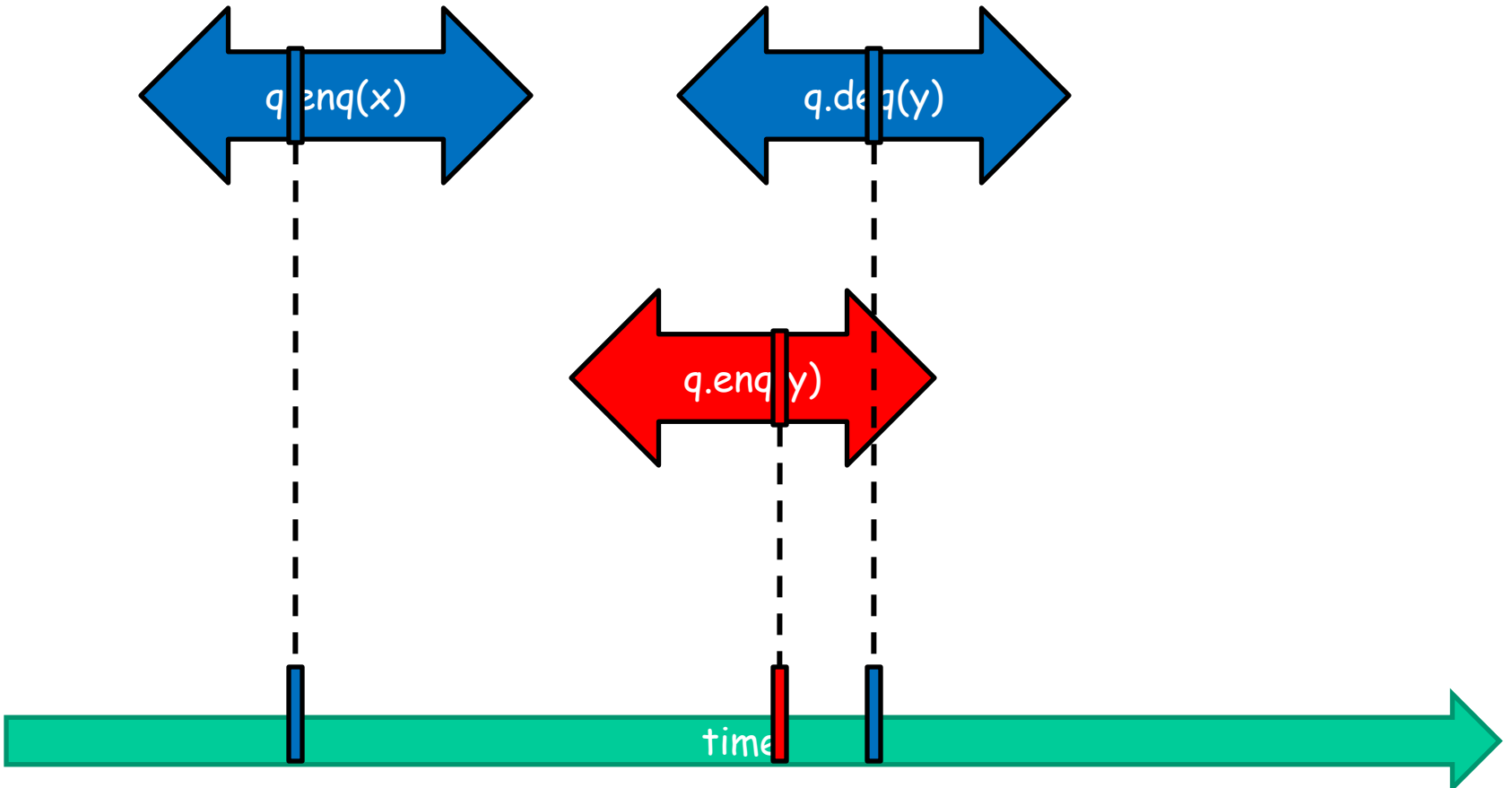
Linearizability

What if we choose other points of linearizability?



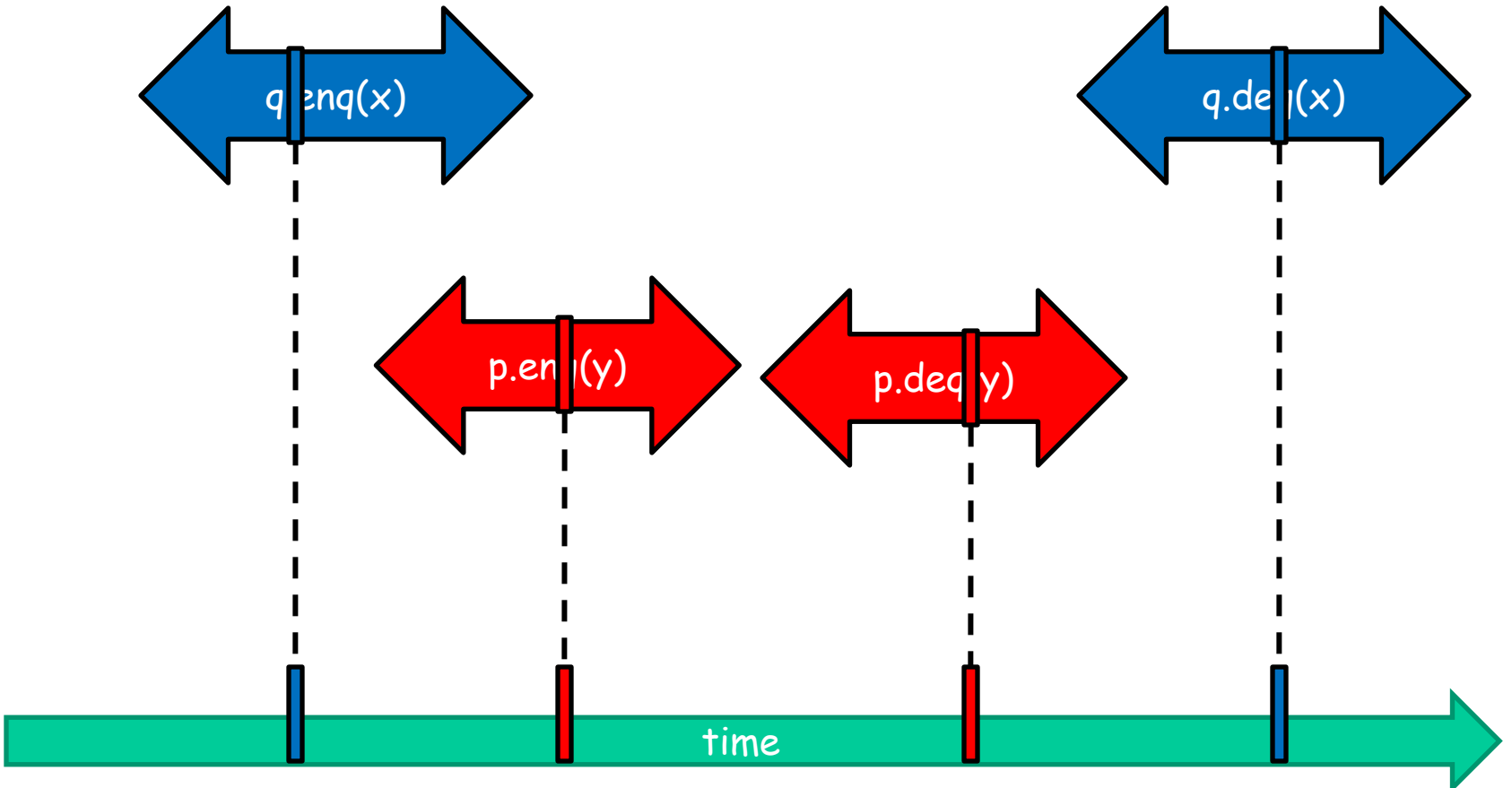
Linearizability

Is this linearizable? **No!**

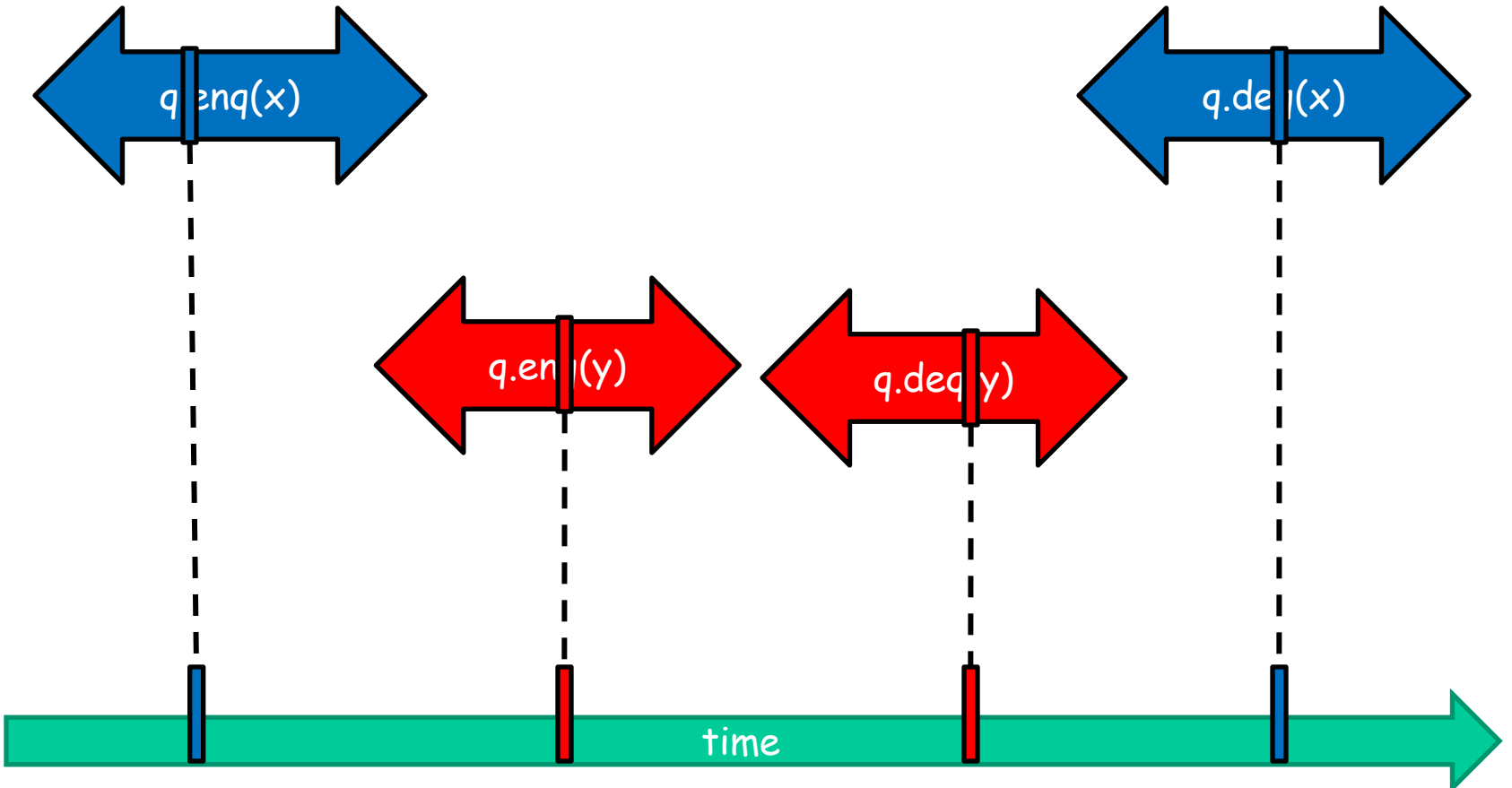


Composition

Is this linearizable? **Yes!**



Composition (same object)



Formal model

- This approach of identifying the atomic step where an operation takes effect ("linearization points") is the most common way to show that an implementation is linearizable.

- In some cases, linearization points depend on the execution.

- We need to define a formal model to allow us to precisely define linearizability (and other correctness conditions).

Formal model

- We split a method call into 2 events:
  - Invocation: method names + args
    - q.enq(x)
  - Response: result or exception
    - q.enq(x) returns void
    - q.deq() returns x or throws emptyException

Formal model

- Invocation notation: *A q.enq(x)*
  - A – thread
  - q – object
  - enq – method
  - x – arg
- Response notation: *A q*: void , *A q*: empty()
  - A – thread
  - q – object
  - void – result, exception

History

A sequence of invocations and responses. It describes an execution.

H =

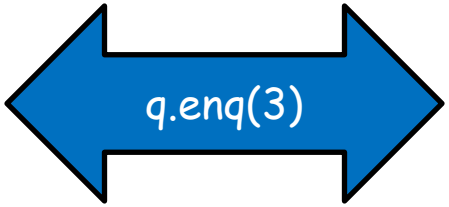
A q.enq(3)
A q:void
A q.enq(5)
B p.enq(4)
B p:void
B q.deq()
B q:3

Definitions

- Invocation and Response match if: thread names and object names agree.

A q.enq(3)
A q:void

And this is what we used before as:



Definitions

- Object projection:

H| q=

A q.enq(3)
A q:void
A q.enq(5)
B q.deq()
B q:3

- Thread projection:

H| A =

A q.enq(3)
A q:void
A q.enq(5)

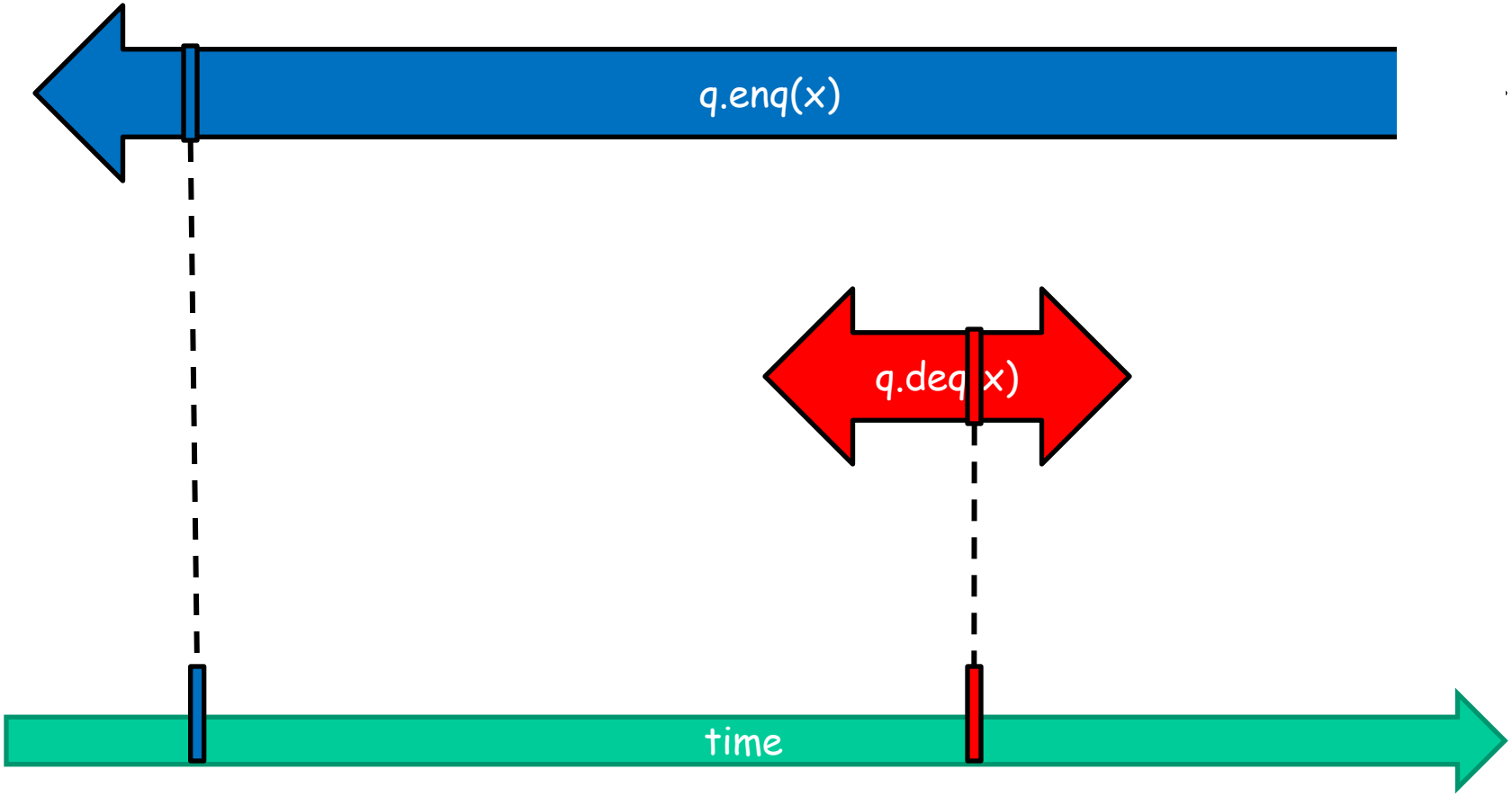
Definitions

- A pending invocation is an invocation that has no matching response.

A q.enq(3)
A q:void
A q.enq(5)
B q.deq()
B q:3

- Complete history: history without pending invocations.

Extending Histories



Definitions

- Sequential history: A sequence of matches, can end with pending invocation.

A q.enq(3)
A q:void
B p.enq(4)
B p:void
B q.deq()
B q:3
A q:enq(5)

Definitions

- Well-formed history: for each thread A, H|A is sequential.
- Equivalent histories: H and G are equivalent if for all threads A: H|A = G|A

H =

A q.enq(3)
B p.enq(4)
B p:void
B q.deq()
A q:void
B q:3

G =

A q.enq(3)
A q:void
B p.enq(4)
B p:void
B q.deq()
B q:3

Definitions

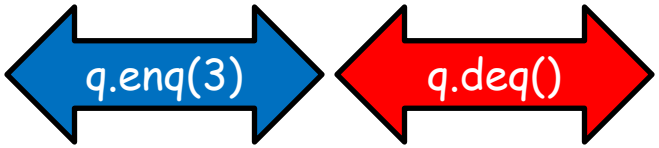
- A method call precedes another if response event precedes invocation event.

A q.enq(3)
B p.enq(4)
B p.void
A q:void
B q.deq()
B q:3

Notation:

$m_0 \rightarrow_H m_1$

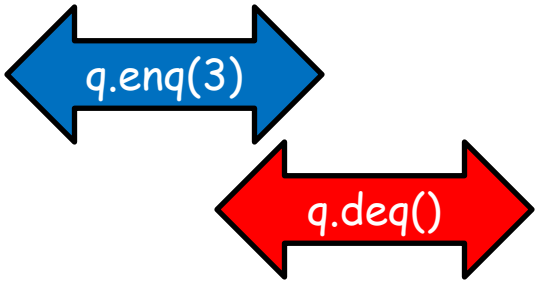
m_0 precedes m_1
(it defines partial order)



Definitions

- Methods can overlap

A q.enq(3)
B p.enq(4)
B p.void
B q.deq()
A q:void
B q:3



Sequential Specifications

- This is a way of telling if single-thread, single-object history is **legal**.
- We saw one technique:
  - Pre-conditions
  - Post-conditions
- but there are more.

Legal history

- A sequential history H is legal if:
  - for each object x, H|x is in the sequential specification for x.
- for example: objects like queue, stack

Linearizability - formally

- History H is linearizable if it can be extended to history G so that G is equivalent to legal sequential history S where $\rightarrow_G \subset \rightarrow_S$.

- G is the same as H but without pending invocations:
  - append responses to pending invocations.
  - discard pending invocations.

Linearizability - formally

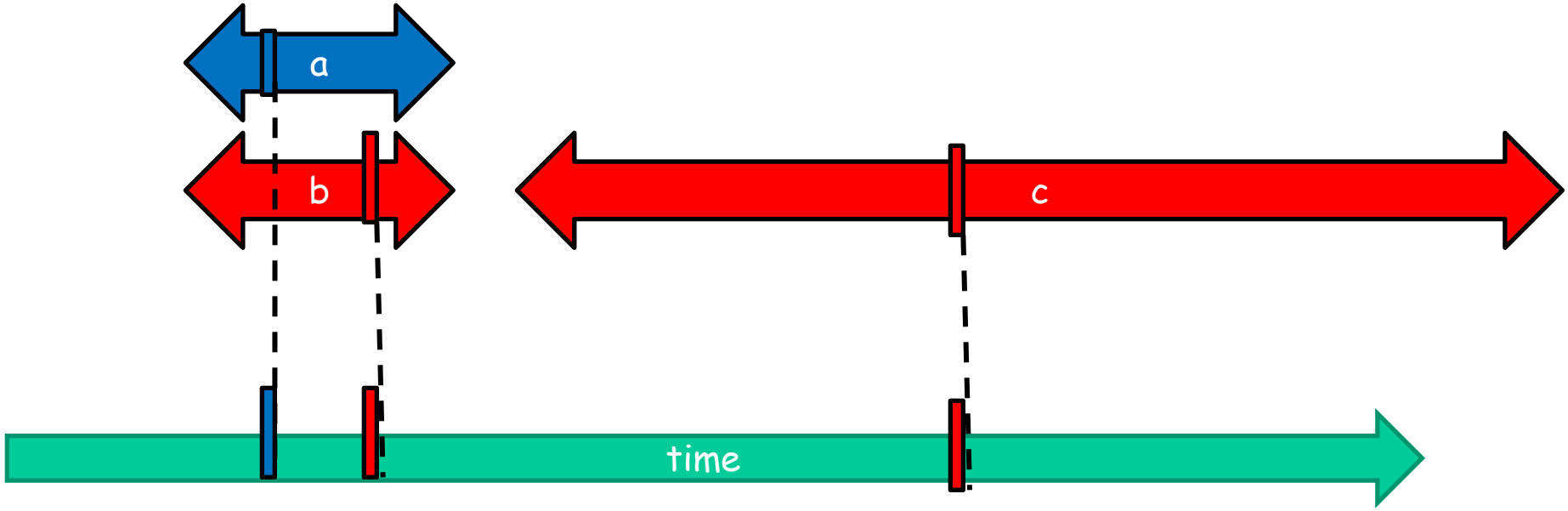
Let's explain what is $\rightarrow_G \subset \rightarrow_S$.

Example:

$\rightarrow_G = \{a \rightarrow c, b \rightarrow c\}$

$\rightarrow_S = \{a \rightarrow b, a \rightarrow c, b \rightarrow c\}$

closure



Example:

Add response to this pending invocation:

Discard this pending invocation:

H =

A q.enq(3)

B q.enq(4)

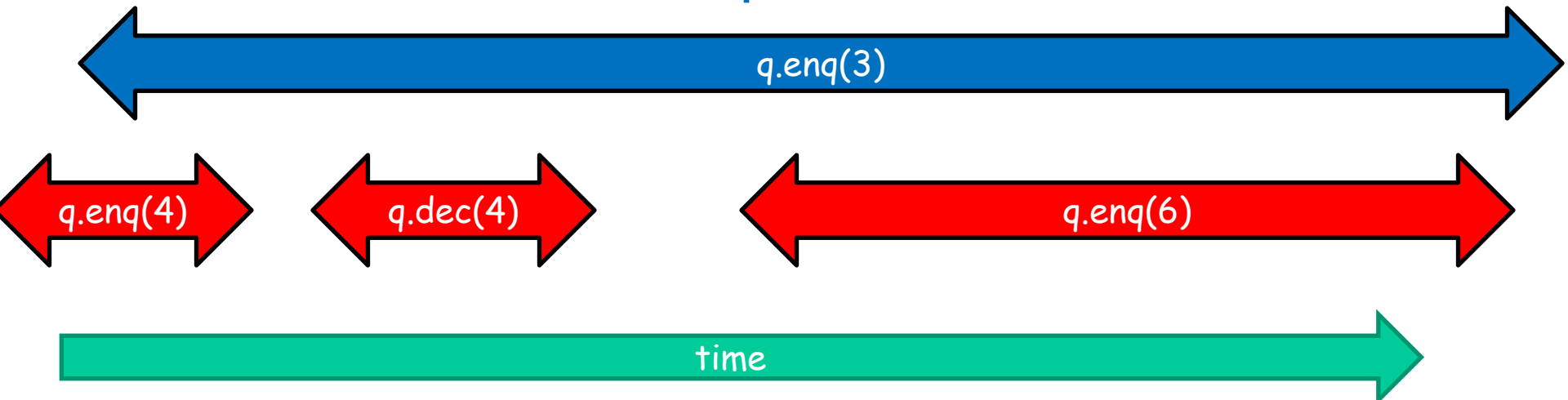
B q:void

B q.deq()

B q:4

B q.enq(6)

A q:void



Example (cont'):

G =
A q.enq(3)
B q.enq(4)
B q:void
B q.deq()
B q:4
A q:void

The equivalent sequential history:

S =
B q.enq(4)
B q:void
A q.enq(3)
A q:void
B q.deq()
B q:4