SALVATORE DI GIROLAMO <DIGIROLS@INF.ETHZ.CH>

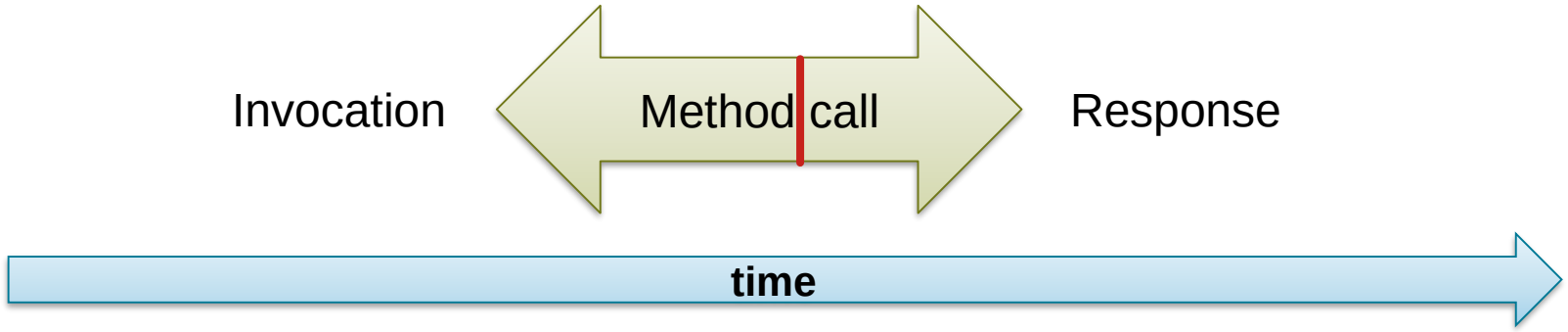
DPHPC: Linearizability

Recitation session

Linearizability vs Sequential Consistency

- **Sequential Consistency?**
  - *Method calls should appear to happen in a one-at-time, sequential order*
  - *Method calls should appear to take effect in program order*

- **Linearizability?**
  - *Method calls should appear to happen in a one-at-time, sequential order*
  - *Each method call should appear to take effect instantaneously at some moment between its invocation and response*



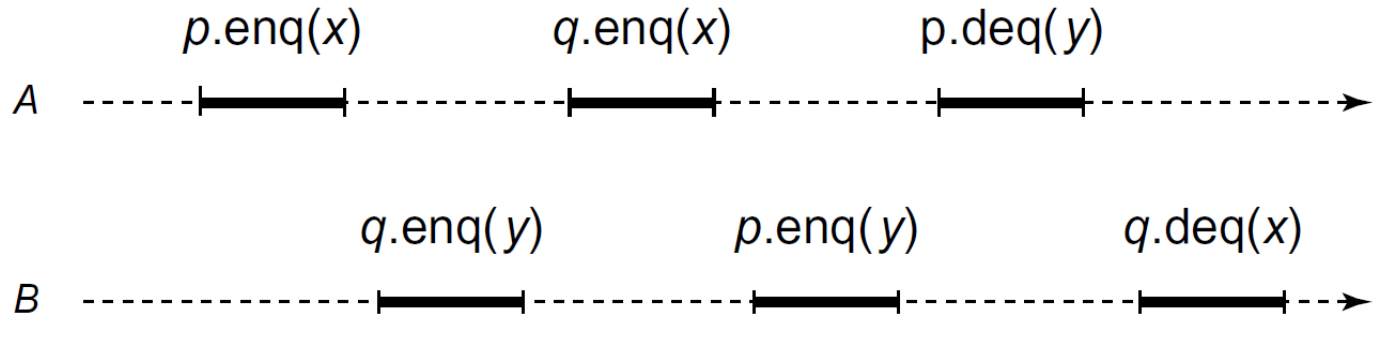
- **Linearizability > Sequential Consistency**
  - Every linearizable execution is sequentially consistent, but not vice versa

Linearizability vs Sequential Consistency

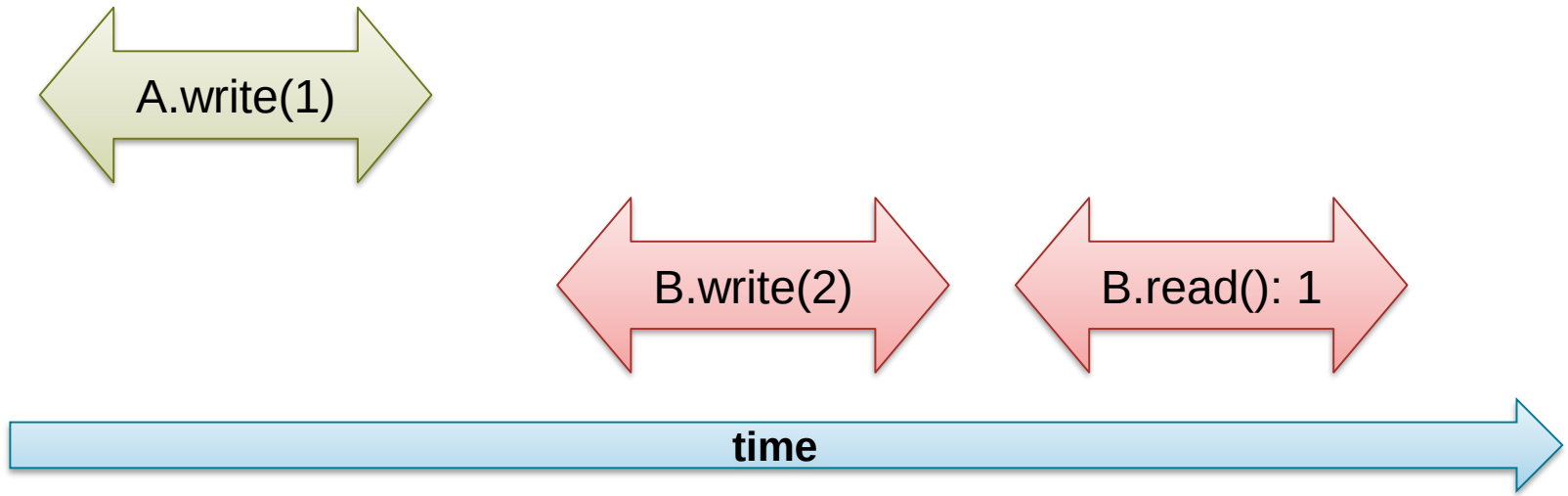
- **Both care about giving an illusion of a single copy**
  - From the outside observer, the system should behave as if there's only a single copy
- **Linearizability cares about time**
- **Sequential consistency cares about program order**

- **Properties of linearizability**
  - Local: A system is linearizable iff each individual object is linearizable.
    Compositional
  - Non-blocking: one method is never forced to wait to synchronize with another
    Does not impact on concurrency

SC is not compositional!



Linearizability vs Sequential Consistency



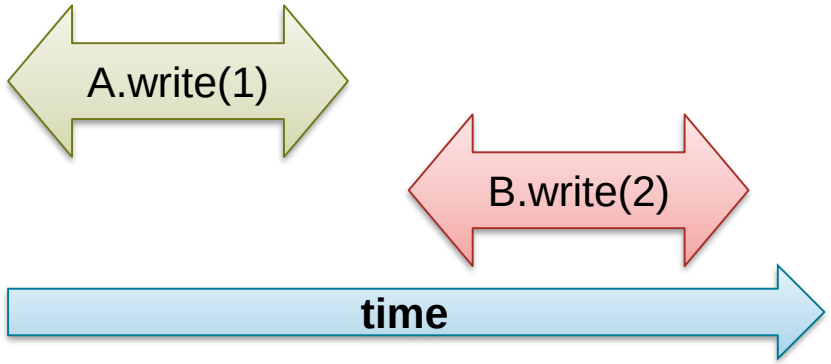
- **Is it sequentially consistent?**
  - Yes, we can reorder B.write(2) and A.write(1)
- **Is it linearizable?**
  - No, the method can “happen” only between its invocation and response

Quiz

- **History**
  - A finite sequence of method invocation and response events
- **Thread projection – H|A**
  - Subsequence of all events in H whose thread names are A
- **Sequential history**
  - The first event is an invocation
  - Each invocation, except possibly the last, is immediately followed by a matching response
- **Concurrent history**
  - Methods can overlap
- **Well-formed history**
  - If each thread subhistory (thread projection) is sequential

Linearizability – Formal definition

- **A method call m0 precedes a method call m1 in history H if m0 finishes before m1 started**
  - m0's response events occurs before m1's invocation event



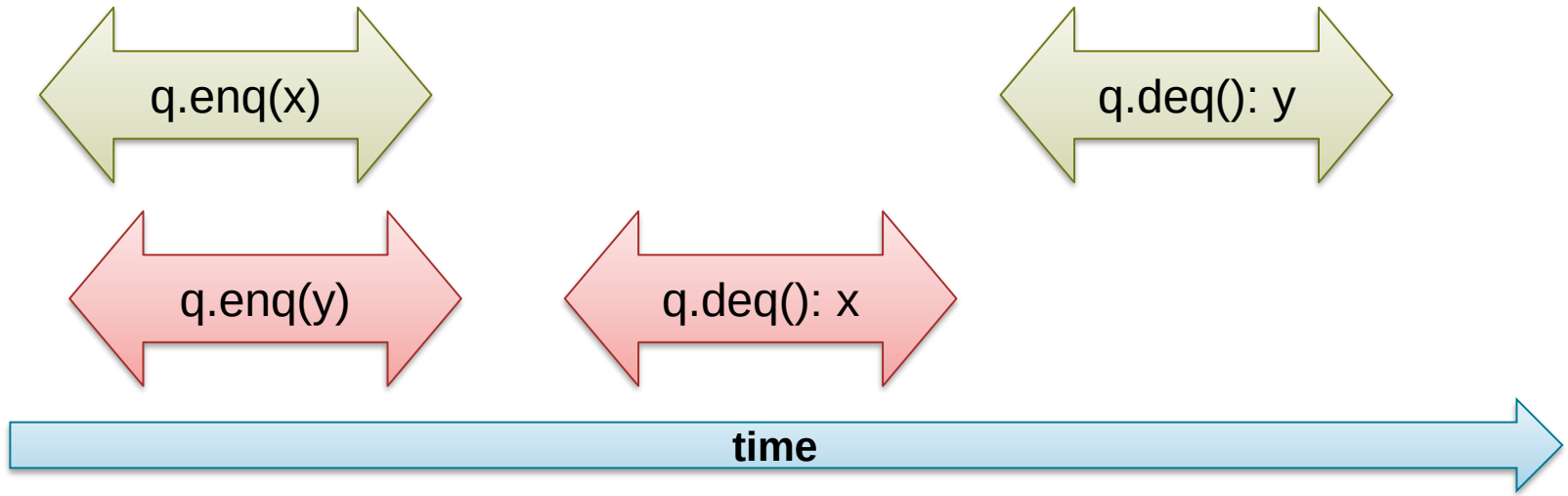
- **Two histories H and H' are equivalent if for every thread A, H|A = H'|A**
- **Given an history H, an extension of H is an history constructed by appending response to zero or more pending invocation in H**
- **Given an history H, complete(H) is the subsequence of H consisting of all the matching invocations and responses (i.e., we remove pending invocations).**
- **A sequential history H is legal if each object subhistory is legal for that object**

Linearizability – Formal definition

- **A history H is linearizable if it has an extension H' and there is a legal sequential history S such that**
  - **L1:** complete(H') is equivalent to S
  - **L2:** if method call m0 precedes method call m1 in H, then the same is true in S

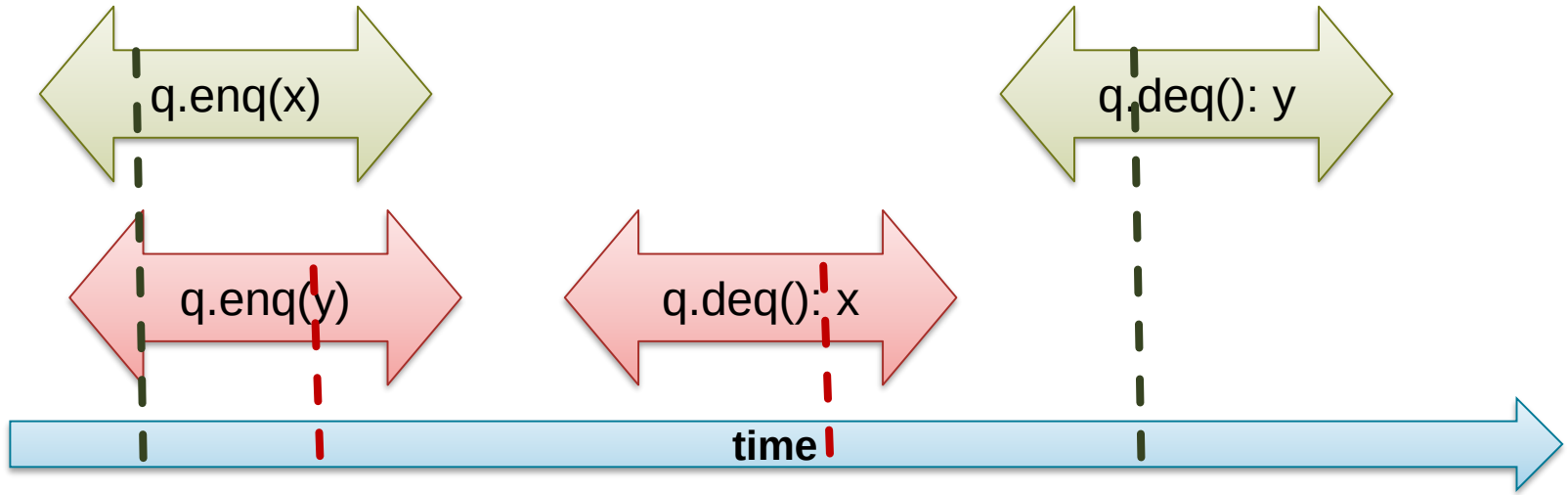
- **If two method calls overlap, we are free to order them in any convenient way**
  - By setting the linearization point

Examples



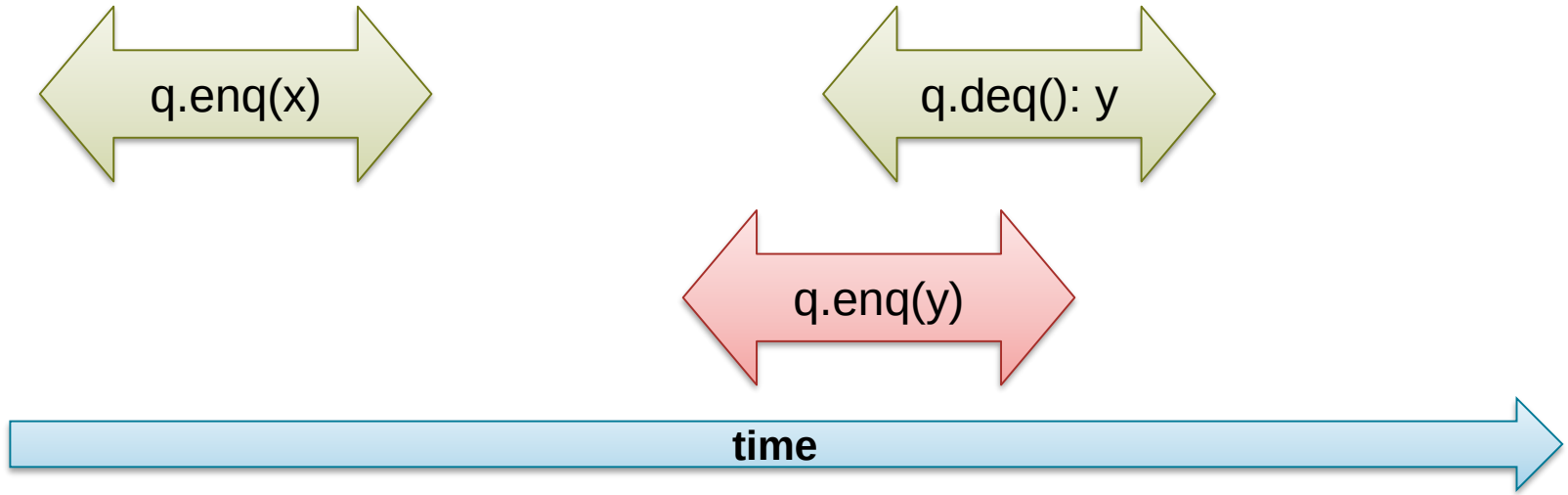
Is this linearizable?

Examples



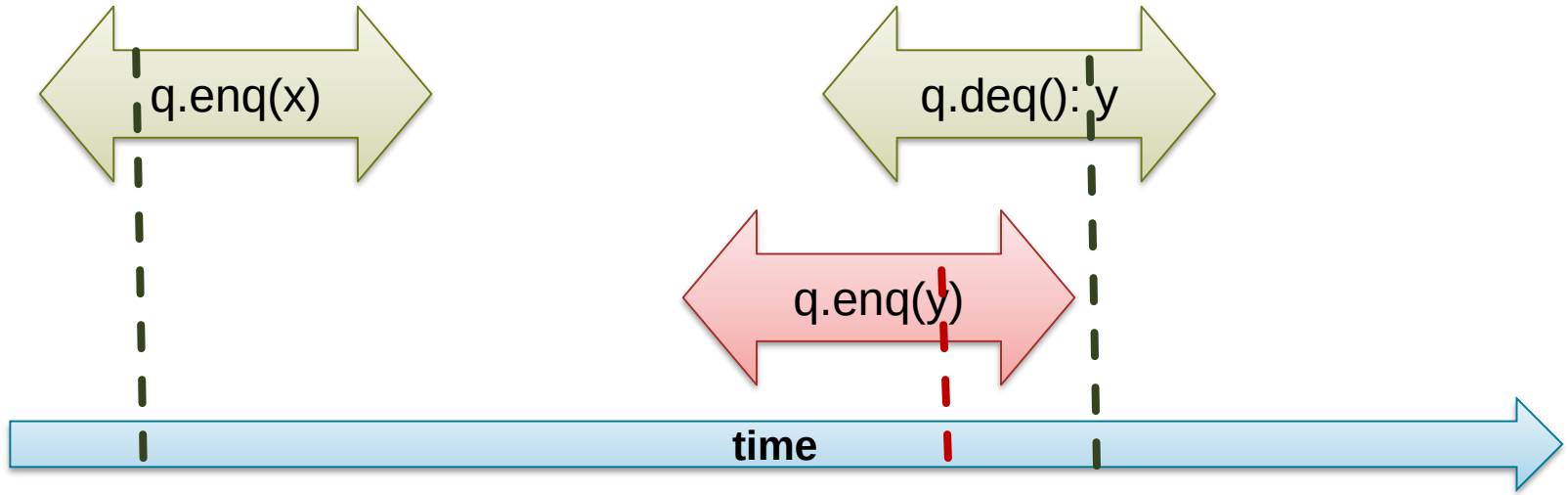
Is this linearizable? Yes

Examples



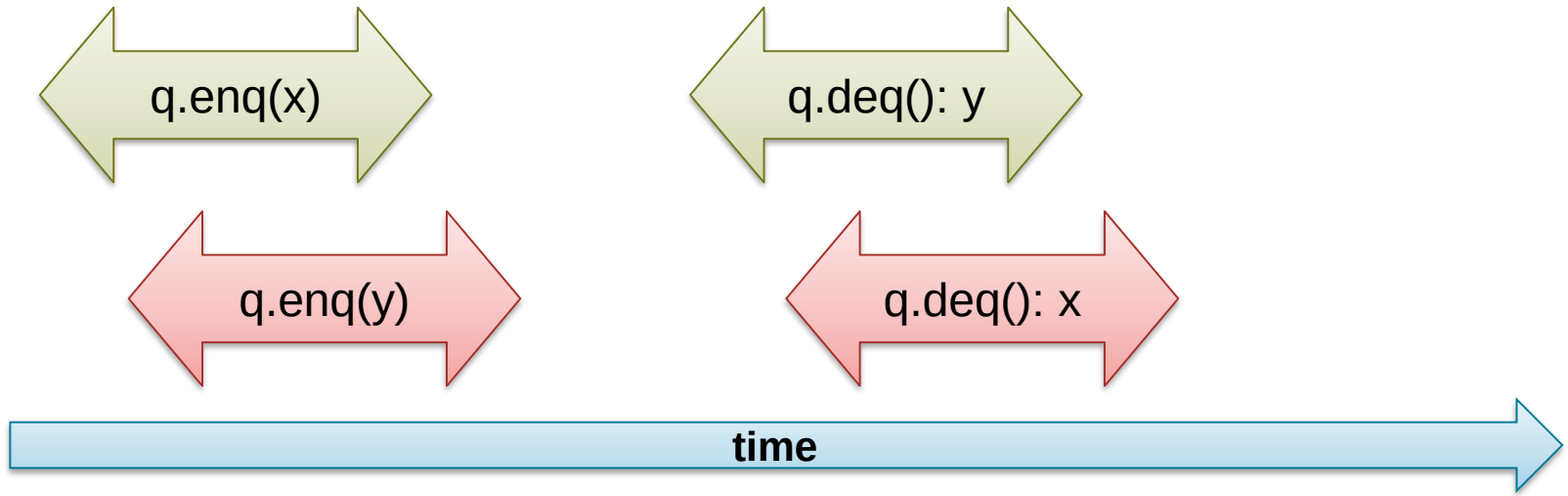
Is this linearizable?

Examples



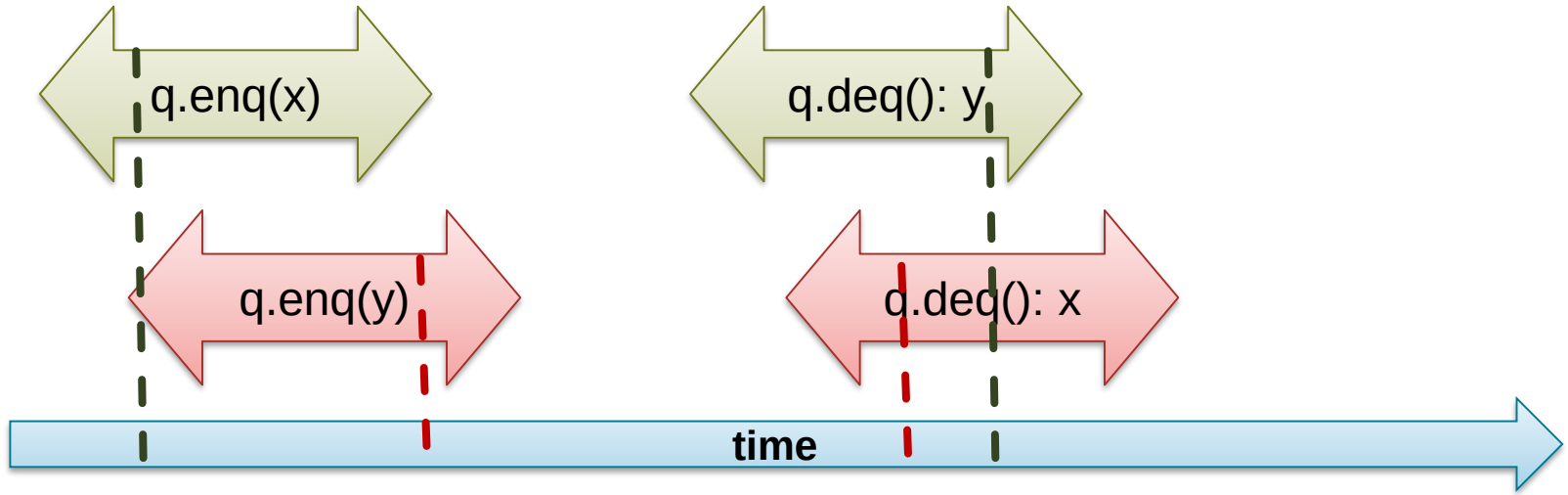
Is this linearizable? No

Examples



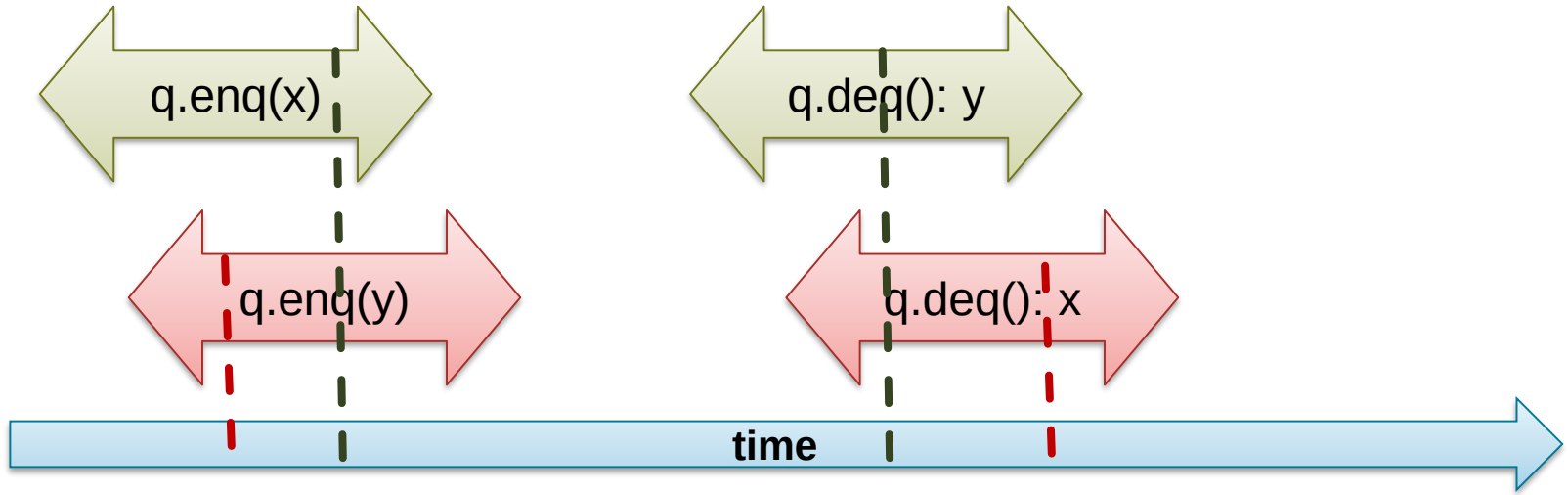
Is this linearizable?

Examples



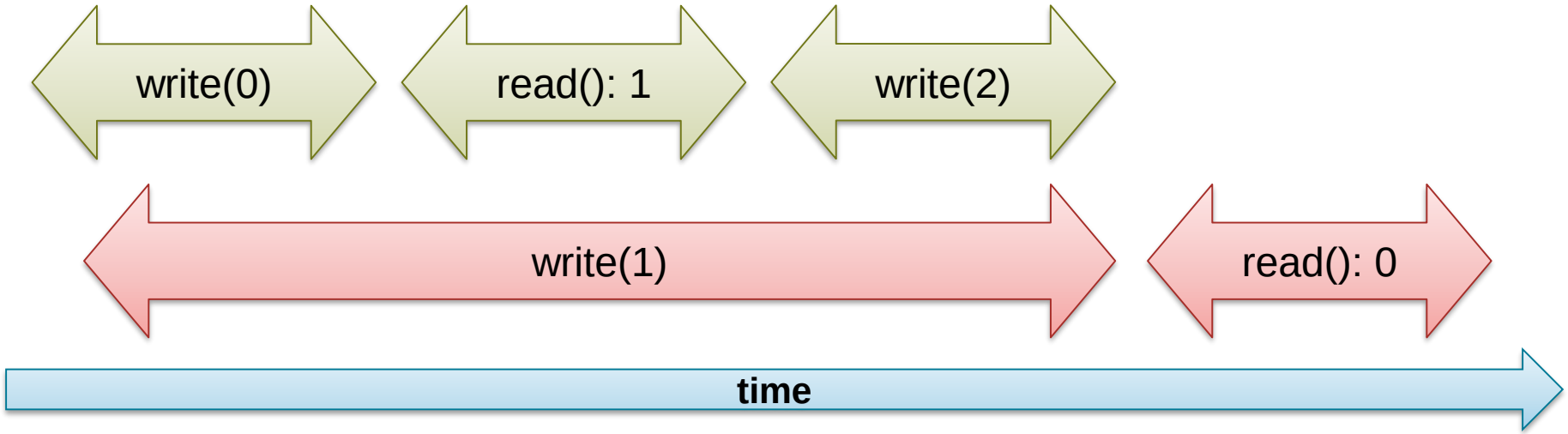
Is this linearizable? Yes

Examples



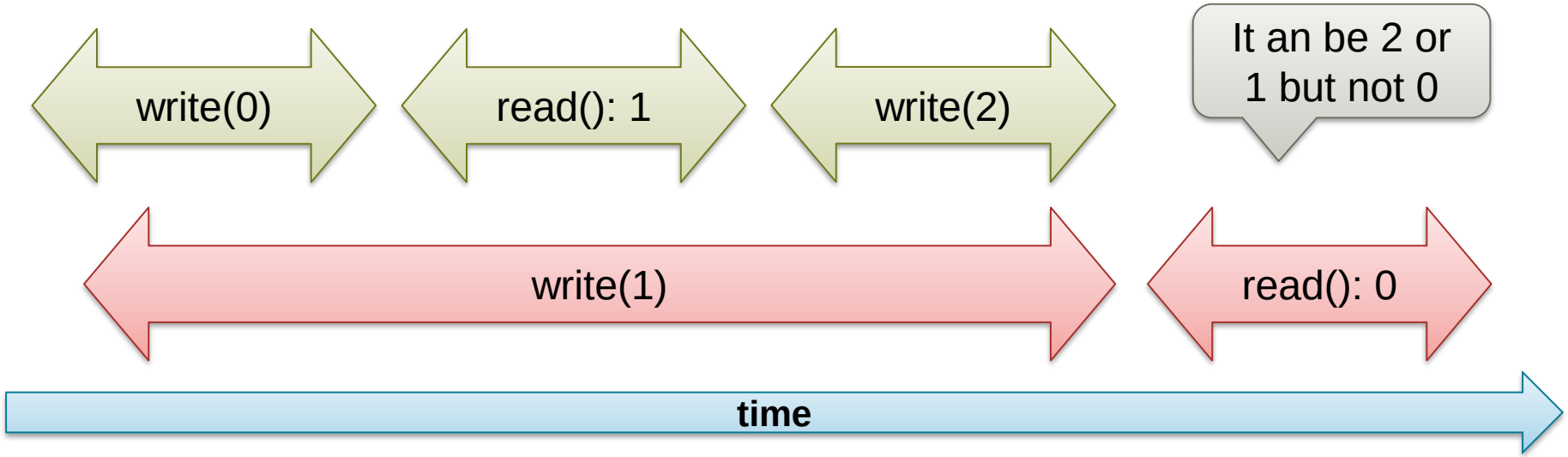
Is this linearizable? Yes, multiple orders

Examples



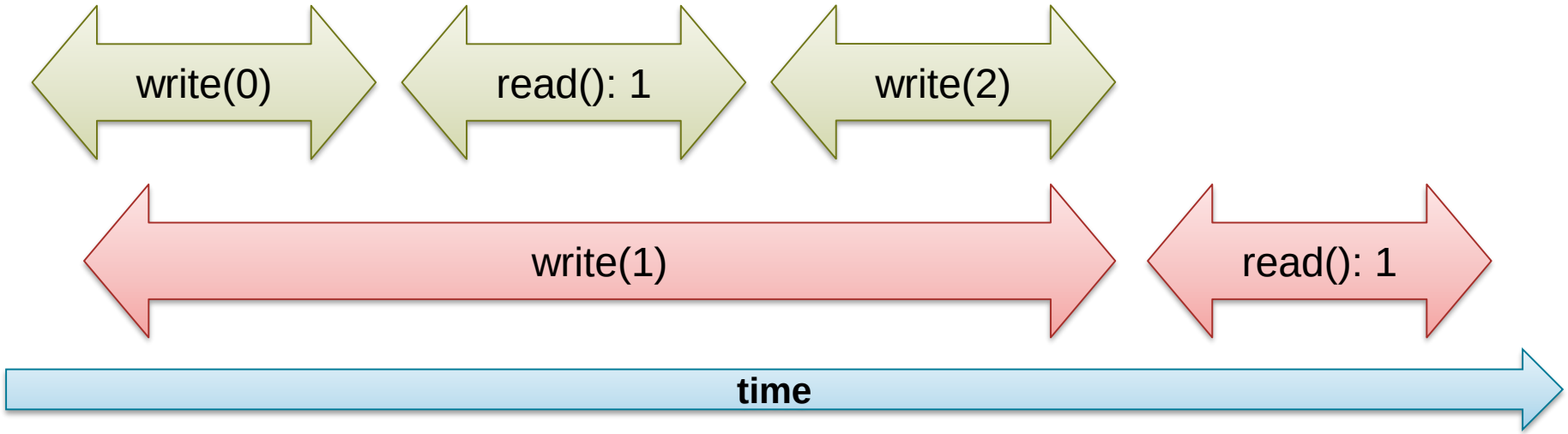
Is this linearizable?

Examples



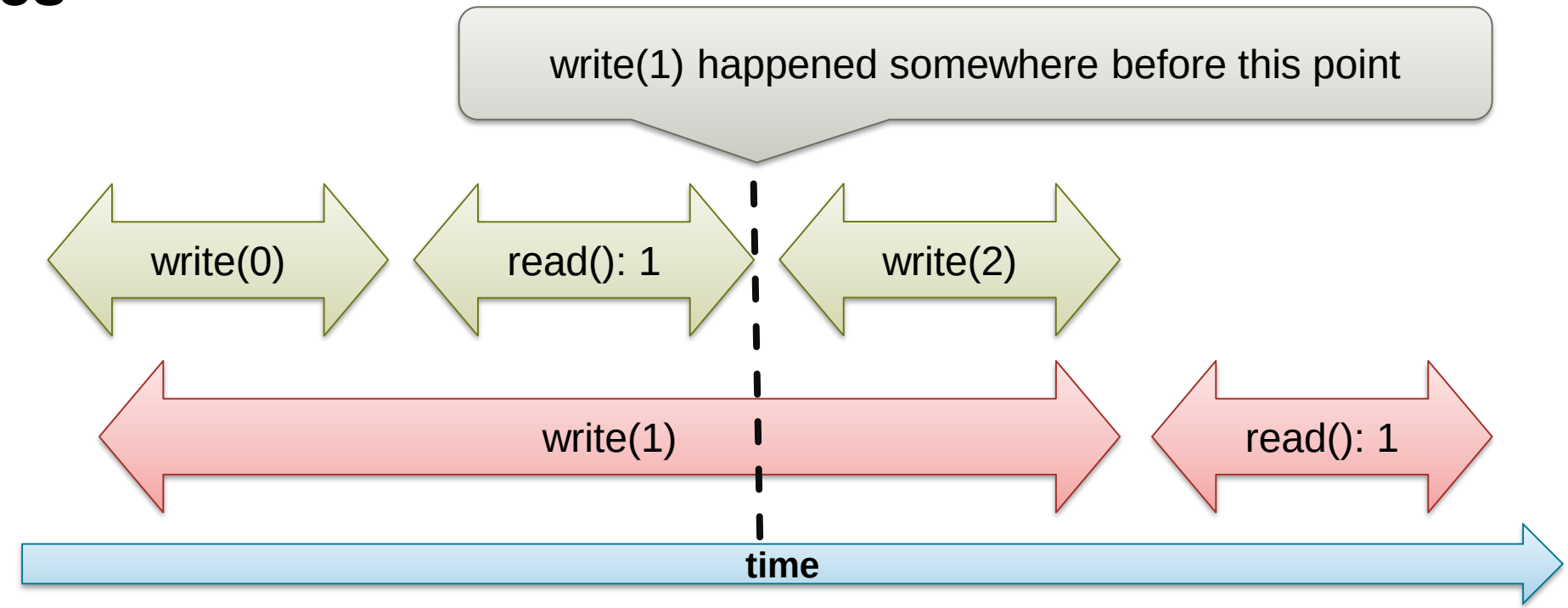
Is this linearizable? No

Examples



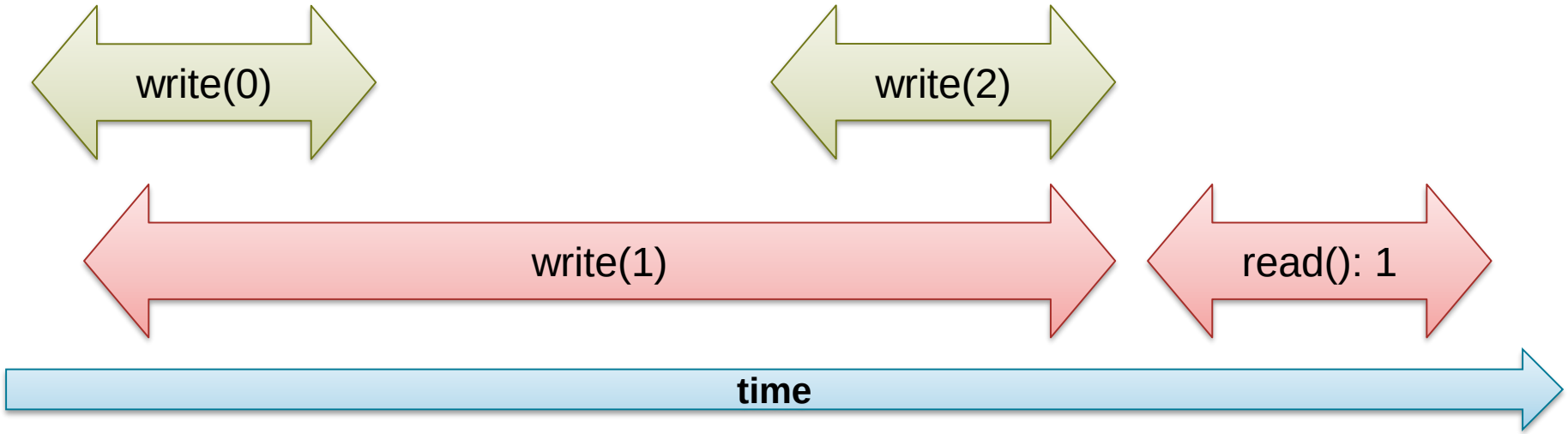
Is this linearizable?

Examples



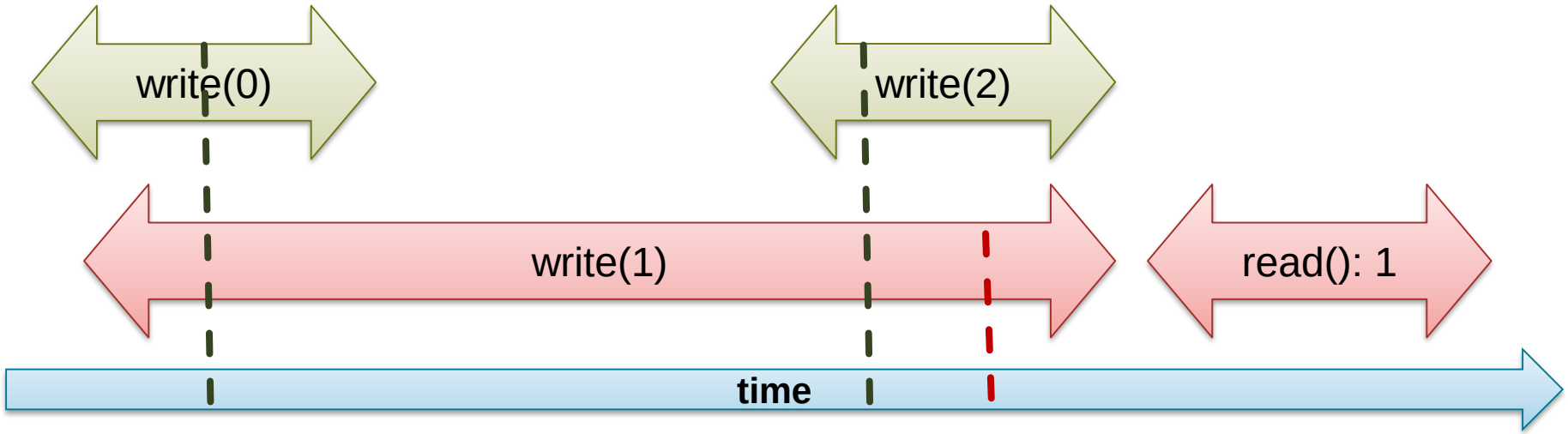
Is this linearizable? No

Examples



Is this linearizable?

Examples



Is this linearizable? Yes