Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2017 Prof. R. Wattenhofer

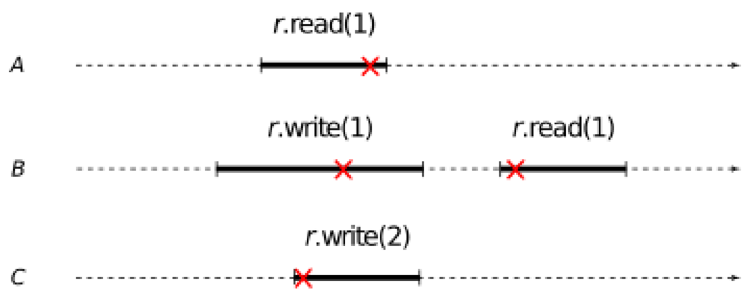
Distributed Systems Part II

Solution to Exercise Sheet 11

Quiz

1 Quiz

a) Yes, the linearization points are marked with a cross.



b) If a thread completes an infinite number of method calls in an infinite history H, then each method call takes a finite number of steps in H. As this is true for every history H, *every* method call finishes in a finite number of steps and x is wait-free.

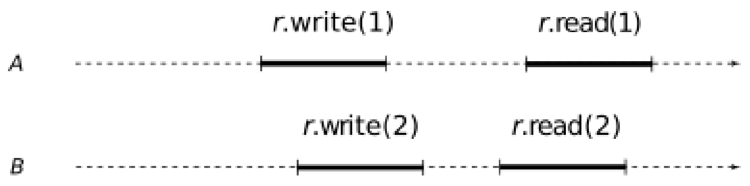
For the other direction, if every method call finishes in a finite number of steps, then an infinite number of method calls are completed in any infinite history H.

c) A large transaction might not fit inside the cache, so some part of the transaction will be evicted from the cache before the transaction is able to complete. Thus, it will be aborted. In such a case, STM or a combination of STM and HTM is helpful.

Advanced

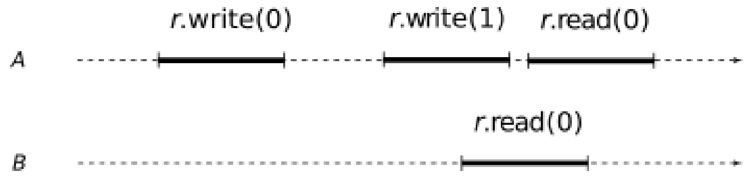
2 Sequential vs. Quiescent

The following execution is sequentially consistent but not quiescently consistent.



The sequential order is r.write(1), r.read(1), r.write(2), r.read(2). There is no quiescently consistent order as there is a quiescence period after both the writes and either the value 1 or 2 can be read afterwards, but not both.

The following execution is quiescently consistent but not sequentially consistent.



It is quiescently consistent because after the quiescent period, the overlapping reads can be ordered before write(1). It cannot be sequentially consistent because thread A reads 0 and there is no write(0) from B that can be ordered after write(1).

3 Linearizability

Consider two processes A and B and the following three operations.

a) $A.enqueue(x)$

b) $B.enqueue(y)$

c) $B.dequeue()$

The first operation runs until just before the last line, that is, it does not put the item x in slot 0. The second operation completes and puts the item y in slot 1. The third operation then starts and finds a null in slot 0. So, it throws an EmptyException(). This shows that the implementation is not linearizable, as in any linearization order, the first dequeued element must be non-empty because there is an element in the queue inserted by the second operation, which is already finished.