Design of Parallel and High Performance Computing
HS 2016
Torsten Hoefler, Markus Püschel
Department of Computer Science
ETH Zurich

Homework 4

Sequential Consistency vs. Linearizability

Please explain the differences between Sequential Consistency and Linearizability.

Solution

In sequential consistency events are only related by program order, i.e., two events that happen in different threads are not related. In linearizability we also require that each method call takes effect at some point between the methods invocation and its response.

To prove either one, we are searching for a total order of the recorded events which satisfies our sequential specification of the object.

Linearizability

Definitions

For the following history of a shared register with the operations write(x)/void and read()/x answer the questions below.

B: r.write(1)
A: r.read()
C: r.write(2)
A: r:1
B: r:void
C: r:void
B: r.read()
B: r:1
A: q.write(3)
C: r.read()
A: q:void

- What is $H|B$?
- What is $H|r$?
- Turn H into a complete subhistory H'.
- Is H' sequential?
- Is H' well-formed?
- Is H' linearizable? If yes, prove it!
- If the first two events are swapped, is the resulting history equivalent to H?

Solution

Thread Subhistory

$H|B$ (H at B) is the thread subhistory of thread B of hitory H. For the given H it is

B: r.write(1)
B: r:void
B: r.read()
B: r:1

Object Subhistory

$H|r$ (H at r) is the object subhistory of object r of history H. For the given H it is

B: r.write(1)
A: r.read()
C: r.write(2)
A: r:1
B: r:void
C: r:void
B: r.read()
B: r:1
C: r.read()

Complete

Add an event C: r:void so that the pending invocation C: r.read() has a matching response.

Sequential

A history H is sequential if:

1. The first event of H is an invocation.
2. Each invocation, except possibly the last, is immediately followed by a matching response. Each response is immediately followed by an invocation.

Note that sequential and sequential consistency are two different things!

Therefore, the given history is not sequential, because the first event is an invocation, but it is followed by another invocation, not by a matching response. A history which is not sequential is concurrent.

Well-formed

A history is well-formed if each thread subhistory $H|P$ of H is sequential. For the given H, the thread subhistories are

A: r.read()
A: r:1
A: q.write(3)
A: q:void

B: r.write(1)
B: r:void
B: r.read()
B: r:1

C: r.write(2)
C: r:void
C: r.read()

All of them are sequential, therefore H is well-formed.

Linearizable

A history is linearizable if it can be reordered into a sequential history, which is correct according to the sequential definition of the object. If a response preceded an invocation in the original history, it must also precede it in the sequential reordering.

Therefore we can reorder the history as shown, which is sequential and observes the semantics of a register (last written value is stored).

C: r.write(2)
C: r:void
B: r.write(1)
B: r:void
A: r.read()
A: r:1
B: r.read()
B: r:1
A: q.write(3)
A: q:void
C: r.read()

Equivalent

Two histories H and H' are eqivalent, if, for every process P $H|P = H'|P$, which is the case if we swap the first two events in the given history.

Overlap

In the following history, do the marked method executions overlap?

A: q.enq(x)
B: q.enq(y)
B: q:void
B: q.deq()
A: q:void
A: q.deq()
B: q:x

Overlap Solution

The method invocations do overlap, as the response **A: q:void** happens after the invocation B: q.deq().

Linearizability, FIFO I

Is the following history of a FIFO queue with the operations enq(x)/void deq()/x linearizable? If yes, prove it! Is it sequentially consistent?

A: r.enq(x)
A: r:void
B: r.enq(y)
A: r.deq()
B: r:void
A: r:y

Linearizability, FIFO Solution I

This history is not linearizable because it violates the semantics of the fifo queue. The enqueueing of x precedes the enqueueing of y, therefore x should also be dequeued first. However, the first dequeue operation retrieves y.

It is sequentially consistent however, as sequential consistency does not require the original event precedence to be preserved.

Linearizability, FIFO II

Is the following history of a fifo queue with the operations enq(x)/void deq()/x linearizable? If yes, prove it!

A: q.enq(x)
B: q.enq(y)
A: q:void
B: q:void
A: q.deq()
C: q.deq()
A: q:y
C: q:y

Linearizability, FIFO II

The given history is not linearizable, because the element y is enqueued only once, but dequeued twice. That violates the semantics of a FIFO queue.