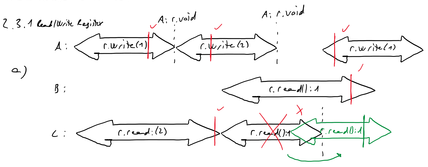
Assignment_2

sabato, 25 marzo 2023 21:22
Marco De Cito 48-760-664

2.3.1 Read/Write Register

a)



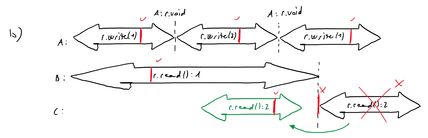
Not Linearizable History (red)
Sequentially Consistent (green)

- This history is not linearizable because the r.read(1) operation of thread C has no chance to read correctly because this event never overlaps with the write(1) event of thread A. However, if we don't consider the intervals themselves, the history is Sequentially Consistent, since we can shift C: r.read(1) to happen later such that we can perform A: r.write(1) before C: r.read():1. To fix the history such that it is linearizable as well, we could provide the following changes:

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

This history solves all problems mentioned above.

b)



Not Linearizable
Sequential consistent (Green)

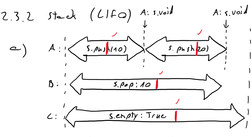
- This history is not linearizable, since C: r.read():2 never overlaps with operation A: r.write(2), so this operation will never be correct. However, this history is Sequentially Consistent, since as shown in green, by ignoring the intervals, we are able to shift operation C: r.read():2 such that it overlaps with operation A: r.write(2) and therefore, read the correct value. The shifting is possible because there are no other events of C which would block the "anticipation" (caused by dependencies). To fix the history such that it becomes linearizable, we can provide the following changes:

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

- This history is both linearizable and Sequential consistent.

2.3.2 Stack (LIFO)

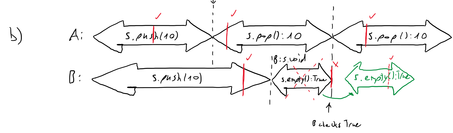
a)



Linearizable
Sequential Consistent

- The stack works with LIFO, therefore, to satisfy Linearizability, push and pop operations of threads A and B must happen before C checks whether s is empty. Also important is that A pushes 20 only after thread C has checked s.empty. So in this case, it is linearizable and also sequential consistent.

b)



Not Linearizable (red)
H is Sequential Consistent (green)

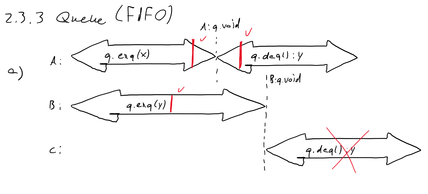
- This history is not linearizable, since B is unable to check whether s is empty as output true. We are sure that B checks "True" until thread A starts its second pop operation, which would leave led B to read True. However, if we not consider the intervals, we are able to shift s.empty():true operation of thread B to be executed later which would solve the previous problem mentioned.

To make this history linearizable as well, we could just postpone B: s: true after A: s.pop() :

A: s.push(10)
B: s.push(10)
A: s.void
A: s.pop()
B: s.void
B: s.empty()
A: s: 10
A: s.pop()
B: s: true
A: s.10

2.3.3 Queue (FIFO)

a)



Not Linearizable
Not Sequential Consistent

- In this history we are able to deq. on y only if Thread B executes enq. operation first and A second. However, even if this is possible, C tries at some point to dequeue another y, which is not possible. Therefore, this history is neither Linearizable nor Sequential consistent, since this problem cannot be fixed by shifting events.

In addition, there is not really a fix in this situation, since changing the event execution order will not solve the problem. The only two options would be to delete one of the two dequeue events of y or substitute one of the two to dequeue x instead.