Assignment 4: Concurrent Objects

ETH Zurich

1 Comparing Histories: Linearizability

Figures 1 and 2 show two different histories for three threads. Each time line corresponds to one thread.

1.1 Task

Decide whether the histories are sequentially consistent. Decide whether the histories are linearizable. Justify your answer.

2 FIFO Queue: Linearizability

The AtomicInteger class is a container for an integer value. One of its methods is boolean compareAndSet(int expect, int update). This method compares the object’s current value to expect. If the values are equal, then it atomically replaces the object’s value with update and returns true. Otherwise, it leaves the object’s value unchanged, and returns false. This class also provides int get() which returns the object’s actual value.

Consider the following FIFO queue implementation. It stores its items in an array items, which, for simplicity, we will assume has unbounded size. It has two AtomicInteger fields. tail is the index of the next slot from which to remove an item. head is the index of the next slot in which to place an item.
class IQueue<T> {
    AtomicInteger head = new AtomicInteger(0);
    AtomicInteger tail = new AtomicInteger(0);
    T[] items = (T[]) new Object[Integer.MAX_VALUE];

    public void enq(T x) {
        int slot = tail.get();
        do {
            slot = tail.get();
        } while (!tail.compareAndSet(slot, slot + 1));
        items[slot] = x;
    }

    public T deq() throws EmptyException {
        T value;
        int slot;

        do {
            slot = head.get();
            value = items[slot];
            if (value == null) {
                throw new EmptyException();
            }
        } while (!head.compareAndSet(slot, slot + 1));
        return value;
    }
}

2.1 Task

Give an example showing that this implementation is not linearizable.