CS1760: Multiprocessor Synchronization Due 8 November 2018

Homework 7

Professor: Maurice Herlihy *TA: Jonathan Lister*

Problem 1. Consider the unbounded queue deq() method shown in Figure 10.7 (reproduced here as Figure 1). Is it necessary to hold the lock when checking that the queue is not empty? Explain.

Problem 2. Consider the linearization points of the enq() and deq() methods of the lock-free queue:

1. Can we choose the point at which the returned value is read from a node as the linearization point of a successful deq()?
2. Can we choose the linearization point of the enq() method to be the point at which the tail field is updated, possibly by other threads (consider if it is within the enq()'s execution interval)? Argue your case.

Problem 3. Consider the unbounded queue implementation shown in Fig. 2. This queue is blocking, meaning that the deq() method does not return until it has found an item to dequeue.

The queue has two fields: items is a very large array, and tail is the index of the next unused element in the array.

1. Are the enq() and deq() methods wait-free? If not, are they lock-free? Explain.
2. Identify the linearization points for enq() and deq(). (Careful! They may be execution-dependent.)

You may assume the array is large enough that we never address an element beyond the array bounds.

```
1  public T deq() throws EmptyException {
2      T result;
3      deqLock.lock();
4      try {
5        if (head.next == null) {
6          throw new EmptyException();
7        }
8        result = head.next.value;
9        head = head.next;
10     } finally {
11       deqLock.unlock();
12     }
13     return result;
14   }
```

Figure 1: The UnboundedQueue class: the deq() method.

```
1  public class HWQueue<T> {
2    AtomicReference<T>[] items;
3    AtomicInteger tail;
4    ...
5    public void enq(T x) {
6      int i = tail.getAndIncrement();
7      items[i].set(x);
8    }
9    public T deq() {
10     while (true) {
11       int range = tail.get();
12       for (int i = 0; i < range; i++) {
13           T value = items[i].getAndSet(null);
14           if (value != null) {
15             return value;
16           }
17       }
18     }
19   }
20 }
```

Figure 2: Queue used in exercise.