CS1760: Multiprocessor Synchronization Due 4 October 2018

Homework 3

Professor: Maurice Herlihy *TA: Jonathan Lister*

Problem 1. Consider a system with an object x and n threads. Determine if each of these properties are equivalent to saying x is deadlock-free, starvation-free, obstruction-free[1], lock-free, wait-free, or none of these. Briefly justify your answers.

1. For every infinite history H of x, an infinite number of method calls are completed.
2. For every finite history H of x, there is an infinite history $H' = H \cdot G$.
3. For every infinite history H of x, every thread takes an infinite number of steps.
4. For every infinite history H of x, every thread that takes an infinite number of steps in H completes an infinite number of method calls.
5. For every finite history H of x, there are n infinite histories $H'_i = H \cdot G_i$ where only thread i takes steps in G_i, where it completes an infinite number of method calls.
6. For every finite history H of x, there is an infinite history $H' = H \cdot G$ where every thread completes an infinite number of method calls in G.

Problem 2. This problem examines a stack implementation (Figure 1) whose push() method does not have a single fixed linearization point in the code.

The stack stores its items in an items array, which for simplicity we will assume is unbounded. The tail field is an AtomicInteger, initially zero. The push() method reserves a slot by incrementing tail, and then stores the item at that location. Note that these two steps are not atomic: there is an interval after tail has been incremented but before the item has been stored in the array.

The pop() method reads the value of tail and then traverses the array in descending order from the tail to slot zero. For each slot, it swaps *null* with the current contents, returning the first non-*null* item it finds. If all slots are *null*, the method returns *null*, indicating an empty stack.

- Give an example execution showing that the linearization point for push() cannot occur at Line 11. Hint: give an execution where two push() calls are not linearized in the order they execute Line 11.
- Give another example execution showing that the linearization point for push() cannot occur at Line 12.
- Since these are the only two memory accesses in push(), we must conclude that push() has no single fixed linearization point. Does this mean push() is not linearizable?

Problem 3. For each of the histories shown below (figures 2 and 3), are they sequentially consistent? Linearizable? Justify your answer.

[1] An implementation is *obstruction-free* when any thread will complete a method call if it is given some finite number of steps in isolation. Note that a step is one of many (possibly infinite) instructions within a method call.

```java
public class AMGStack<T> {
  AtomicReferenceArray<T> items;
  AtomicInteger tail;
  static final int CAPACITY = 1024;

  public AMGStack() {
    items = new AtomicReferenceArray<T>(CAPACITY);
    tail = new AtomicInteger(0);
  }
  public void push(T x) {
    int i = tail.getAndIncrement();
    items.set(i,x);
  }
  public T pop() {
    int range = tail.get();
    for (int i = range - 1; i > -1; i--) {
      T value = items.getAndSet(i, null);
      if (value != null) {
        return value;
      }
    }
    // Return Empty.
    return null;
  }
}
```

Figure 1: Afek/Morrison/Gafni stack.

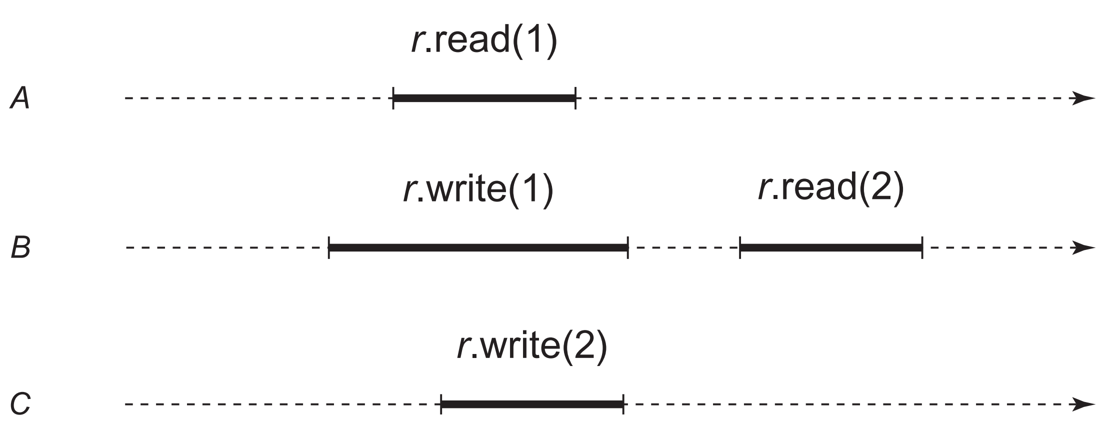

Figure 2: First history for Exercise 3.

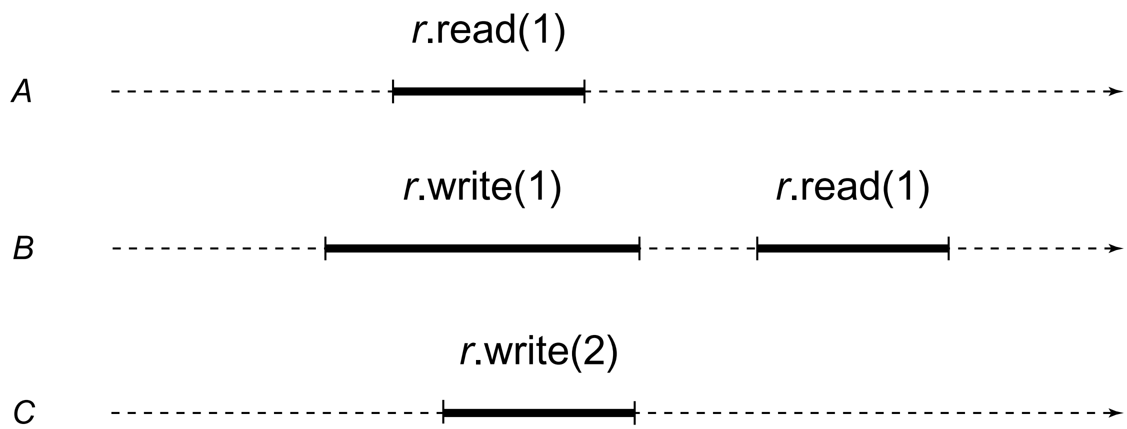

Figure 3: Second history for Exercise 3.