Concurrent Algorithms October 12, 2020

Exercise 2 Solution

Problem 1.

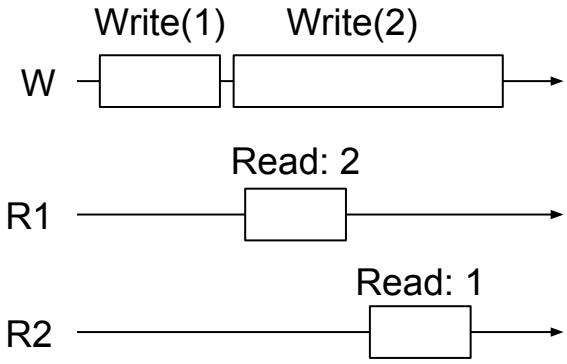

Figure 1: An execution that is possible with a regular register but not with an atomic register. There are three processes: a writer (W) and two readers (R1 and R2). The two reads are concurrent with the second write. The read by R1 completely precedes the read by R2. The execution is not atomic because it is impossible to assign linearization points to all operations: if the linearization point of *Write(2)* is before that of the read by R1, then the read by R2 cannot have a linearization point; if the linearization point of *Write(2)* is after that of the read by R1, then that read cannot have a linearization point.

Problem 2. The transformation does not work for multiple readers (the result is not an atomic register). The non-atomic execution in Figure 1 is possible in this case. Since the register is regular, the read by R1 may read the value 2 being concurrently written by W. Since this is R1's first read operation, the timestamp it obtains for value 2 is higher than its local timestamp. Later, the read by R2 (also concurrent with *Write(2)*) may read the previous value of the register (1). Since this is R2's first read operation, the timestamp it obtains for value 1 is also higher than its local timestamp.

Problem 3. The transformation does not work for multiple writers because each writer has a local timestamp, i.e., the different writers do not share the same time.
In fact, consider two writers W1 and W2. W1 is a more active writer than W2. Assume W1 has already written 10 times to the register (t_1 = 10). After that, W2 writes for the first time to the register, its local timestamp will then be less than 10 (t_2 = 1).
Therefore, any subsequent read after W2's write will miss W2's newly written value, and return the last value written by W1. This violates the sequential property of registers, i.e., a sequential read after a write operation should always return the last written value.