Concurrent Programming, August–November 2016

Assignment 1, 10 September, 2016
Due: 22 September, 2016

Note: Only electronic submissions accepted, via Moodle.

All exercises are from "The Art of Multiprocessor Programming" by Maurice Herlihy and Nir Shavit and the exercise numbers are from the book. Refer to the book if there is any ambiguity.

Chapter 2

Exercise 9 Define r-bounded waiting for a given mutual exclusion algorithm to mean that if $D_j \rightarrow D_k$ then $CS_j \rightarrow CS_{k+r}$. Is there a way to define a doorway for the `Peterson` algorithm such that it provides r-bounded waiting for some value of r?

Exercise 12 Show that the `Filter` lock allows some threads to overtake others an arbitrary number of times.

Chapter 3

Exercise 22 Consider a *memory object* that encompasses two register components. We know that if both registers are quiescently consistent, then so is the memory. Does the converse hold? If the memory is quiescently consistent, are the individual registers quiescently consistent? Outline a proof, or give a counterexample.

Exercise 25 If we drop condition **L2** from the linearizability definition (reproduced below), is the resulting property the same as sequential consistency? Explain.

Definition A history H is *linearizable* if it has an extension H' and there is a legal sequential history S such that

L1 $complete(H')$ is equivalent to S, and

L2 if method call m_0 precedes method call m_1 in H, then the same is true in S.

Chapter 4

Exercise 41 Consider the following implementation of a `Register` in a distributed, message-passing system. There are n processors $P_0, \ldots, P_{n1}$ arranged in a ring, where P_i can send messages only to $P_{i+1 \bmod n}$. Messages are delivered in FIFO order along each link.

Each processor keeps a copy of the shared register.

- To read a register, the processor reads the copy in its local memory.
- A processor P_i starts a `write()` call of value v to register x, by sending the message "P_i: write v to x" to $P_{i+1 \bmod n}$.
- If P_i receives a message "P_j : write v to x," for $i \neq j$, then it writes v to its local copy of x, and forwards the message to $P_{i+1 \bmod n}$.

- If P_i receives a message "P_i: write v to x," then it writes v to its local copy of x, and discards the message. The `write()` call is now complete.

Give a short justification or counterexample.

If `write()` calls never overlap,

- Is this register implementation regular?
- Is it atomic?

If multiple processors call `write()`,

- Is this register implementation atomic?

Chapter 5

Exercise 54 Suppose we augment the FIFO Queue class with a `peek()` method that returns but does not remove the first element in the queue. Show that the augmented queue has infinite consensus number.

Exercise 55 Consider three threads, A, B, and C, each of which has a `MRSW` register, X_A, X_B, and X_C, that it alone can write and the others can read.

In addition, each pair shares a `RMWRegister` register that provides only a `compareAndSet()` method: A and B share R_{AB}, B and C share R_{BC}, and A and C share R_{AC} . Only the threads that share a register can call that registers `compareAndSet()` method or read its value.

Your mission: either give a consensus protocol and explain why it works, or sketch an impossibility proof.