Problem Set 6, Part a

Due: Thursday, December 3, 2015

Readings:

Chapter 13.
Herlihy's paper on the wait-free consensus hierarchy.
Attiya, Welch book, Chapter 15.

Next week:
Borowsky, Gafni, Lynch, Rajsbaum paper.
Attiya, Welch book, Section 5.3.2
Attie, Guerraoui, Kouznetsov, Lynch, Rajsbaum paper
Lynch book, Chapter 17
Lamport's "Part-Time Parliament" paper

Problems:

1. Exercise 13.11.

   Consider a modification of the *UnboundedSnapshot* algorithm in which each *snap* and *embedded-snap* looks for three different tags for some *x(i)* rather than four as described. Is the modified algorithm still correct? Either prove that it is or give a counterexample execution.

2. Exercise 13.17.

   Give a simplified version of the *Bakery* algorithm of Section 10.7 that uses snapshot shared variables. Prove its correctness.

3. Exercise 13.24.

   Design a simplified version of the *VitanyiAwerbuch* algorithm for the setting where the read/write shared variables are single-writer/multi-reader variables. Is the propagation phase of the *READ* protocol needed? Prove correctness and analyze complexity.

4. This exercise is about determining the consensus number (defined in Herlihy's paper and in class) of a "fetch-and-increment counter" variable type. An object of this type starts with an initial value of 0. The only operation is "fetch-and-increment", which increments the counter value and returns the previous value.

   (a) Give a formal definition of a "fetch-and-increment counter" variable type, using the notation given in Section 9.4 for variable types.

   (b) What is the consensus number of the fetch-and-increment variable type? That is, what is the largest number n for which fetch-and-increment shared objects, plus shared read/write registers, can be used to solve n-process consensus?

   (c) For your n from part (b), describe an n-process consensus algorithm using fetch-and-increment shared objects, plus shared read/write registers.

   (d) Prove that no combination of fetch-and-increment shared objects and registers are sufficient to solve $n + 1$-process consensus.

(e) How do your answers change if, instead of a fetch-and-increment counter, we use a simple counter, with two kinds of operations, increment and read?