Problem Set 6, Part b

Due: Thursday, December 3, 2009

Reading:

Herlihy paper on wait-free synchronization.
(Optional) Attiya, Welch, Chapter 15.

Reading for next week:

Borowsky, Gafni, Lynch, Rajsbaum paper.
Attiya, Welch, Section 5.3.2 (optional).
Attie, Guerraoui, Kouznetsov, Lynch, Rajsbaum paper on boosting fault-tolerance.
Chapter 17 of Lynch book.
Lamport's "Part-Time Parliament" paper.

Problems:

1. Herlihy's paper contains an algorithm, covered in class, that shows how to implement 2-process wait-free consensus using queue objects. However, the queue objects used in the algorithm are initialized by enqueueing the value 0 and then the value 1. Describe a new algorithm that uses initially-empty queues.

2. This exercise is about determining the consensus number (defined in Herlihy's paper and in class) of the "stack" variable type.

   (a) Give a formal definition of the "stack" variable type (see Section 9.4 for the notation we use for variable types). It should have two operations, push and pop. The pop operation should return, and remove, the last item pushed onto the stack, from among those still remaining on the stack. If the stack is empty, the pop operation should return a special "empty" indicator.

   (b) Prove that the consensus number of your stack variable type is at least two.

   (c) Prove that the consensus number of your stack variable type is at most two.

3. In class, we described a simplified version of Herlihy's universality construction for wait-free consensus objects. It implemented an arbitrary n-process atomic object using an infinite sequence of n-process consensus objects to decide on successive operations, and an "announce" array of read/write registers to ensure fairness.
   Describe (in clear English) a mechanism that you can add to this algorithm to try to achieve a good time upper bound for all operations. Try to achieve $O(n\ell)$, where ℓ is an upper bound on process step time.

4. Consider the safe agreement protocol used in the BG simulation, described in the journal paper by Borowsky, Gafni, Lynch, and Rajsbaum and covered in class. In that algorithm, the processes interact using snapshot shared memory.
   Now, suppose that the algorithm is rewritten to use single-writer multi-reader read/write shared registers instead of snapshot memory. That is, the algorithm works the same as before, except that, instead of taking a snapshot, a process simply reads all the processes' registers, one at a time, in a series of separate steps.

(a) Does the resulting algorithm still yield safe agreement?

(b) If your answer to part (a) is "yes", then sketch the key steps of a modified proof. If your answer is "no", exhibit a counterexample execution.

6.852J / 18.437J Distributed Algorithms

Fall 2009