Problem Set 5, Part b

Due: Thursday, November 19, 2015

Readings:

Chapter 12
Sections 13.1-13.2.
Next week: Rest of Chapter 13
Herlihy, Wait-free synchronization
Attiya, Welch book, Chapter 15

Problems:

5. (Exercise 12.5.) Reconsider the agreement problem using read/write shared memory. This time consider a more constrained fault model than general stopping failures, in which processes can fail only at the very beginning of computation. Can the agreement problem be solved in this model, guaranteeing

   (a) 1-failure termination?

   (b) wait-free termination?

6. Consider the agreement problem using a combination of read/write registers and *multiset variables*. Each value taken on by a multiset variable is a finite multiset of elements of some basic set S, that is, a finite set of elements of S together with a positive integer *multiplicity* for each element. The initial value of a multiset variable is the empty multiset.

   A multiset variable supports two kinds of operations: $add(s)$, $s \in S$, which adds one copy of the element s to the multiset, that is, it increases s's multiplicity by 1, and $number(s)$, which does not change the multiset but returns the multiplicity of s (0 if s is not in the multiset).

   (a) Write the variable type definition for a multiset variable carefully, following the style used in Section 9.4.

   (b) Prove that the n-process agreement problem with wait-free termination cannot be solved using any finite number of shared variables, where each variable is either a read/write register or a multiset variable.

   (c) Same question as part (b), but for 1-failure termination.

7. Give an algorithm to solve 3-consensus for $n > 3$ processes, satisfying 2-failure termination. Your algorithm should work in the asynchronous shared-memory model with read/write registers.

8. Suppose that we modify Example 13.1.4 so that the system supports $add2$ operations as well as *read* and *increment*. The algorithm is the same as before, with the following addition: when an $add2_i$ input occurs on port i, process i adds 2 to $x(i)$.

   Is the resulting system a read/increment/add2 atomic object? Either prove that it is or give a counterexample execution.