Concurrent Algorithms December 1, 2015

Exercise 9

Problem 1. Prove the correctness of the Adopt-Commit implementation from the lecture.

Problem 2. Prove the correctness of the adopt-commit-based consensus from the lecture in the two following cases:

a) When all processes verify $leader_i = i$ forever. The algorithm is only *obstruction-free* in this case.

b) When there is a correct process such that, eventually, any correct process p_j verifies $leader_j = i$ forever. The algorithm is then *wait-free*.

Problem 3. A *k-set-agreement* object is a generalization of a consensus object in which processes could decide up to k different values. Formally, k-set-agreement satisfies the following properties:

1. *Validity:* Values decided by each process are the values proposed some processes.
2. *Agreement:* At most k different values could be decided.
3. *Termination:* Every correct process eventually decides a value.

Your task is to show that k-set-agreement and k-consensus (or k-simultaneous agreement), given in the class, are equivalent. That is, you have to show that one implements the other.

Hint: When implementing k-consensus using k-set-agreement, an algorithm that solves the problem is the following:

1: **function** KSC.PROPOSE($v_1, \ldots, v_k$)
2: $\quad V_i \leftarrow [v_1, \ldots, v_k]$
3: $\quad dV_i \leftarrow kSA.\text{PROPOSE}(V_i)$
4: $\quad REG[i] \leftarrow dV_i$
5: $\quad snap_i \leftarrow REG.\mathsf{snapshot}()$
6: $\quad c_i \leftarrow$ number of distinct (non-$\bot$) vectors in $snap_i$
7: $\quad d_i \leftarrow$ minimum (non-$\bot$) vector in $snap_i$
8: $\quad \mathsf{return}\langle c_i, d_i[c_i]\rangle$
9: **end function**

Where $REG[0, \ldots, n-1]$ in an array of single-writer multi-readers atomic registers initialized at $\bot$. Processes write atomically a *vector of values* in their register (Line 4). $REG.\mathsf{snapshot}()$ returns an atomic snapshot of this array of registers. Consequently, $snap_i[0, \ldots, n-1]$ is an array of vectors, possibly containing $\bot$ values for some indices. We suppose that there is an order on the set of values that can be proposed, and we use the induced *lexicographic order* on vectors at Line 7.

Your task is then to prove that the algorithm implements a k-simultaneous consensus from k-set agreement objects and atomic registers.

Problem 4. Below is an algorithm that implements a single state machine replication using consensus shared objects:

Local:

sM // a copy of the state machine
Commands // a list of command
ready // binary register (initially true)

Shared:

Consensus // a list of shared consensus objects

```
while(true) {
   if ready then c = Commands.next()
   cons = Consensus.next()
   c’ = cons.propose(c)
   sM.perform(c’)
   if c’ == c then ready = true
   else ready = false
}
```

The algorithm ensures the following correctness properties:

1. *Validity:* If a process p_i performs command c, then c was issued by some process p_j and p_i performed every command issued by p_j before c.
2. *Ordering:* If a process performs command c without having performed command c', then no process performs c' without having performed c.
3. *Progress:* Every correct process performs an infinite number of commands on the state machine.

However the algorithm is not *fair*, i.e. it does not ensure the following property:

- *Fairness:* If a correct process issues command c, then it eventually performs c on the state machine.

Your task:

1. Show why the algorithm does not ensure fairness, i.e. show an execution violating the property.
2. Modify the algorithm so that the resulting algorithm would ensure fairness.