Concurrent Algorithms 2013
Exercise 9

November 20, 2013

Problem 1

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, given in the class, are equivalent. That is, you have to show that one implements the other.

Problem 2

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.