Concurrent Computing October 14, 2025

Exercise 4

Problem 1.

A *binary consensus* shared object has a single operation *propose* that takes a value v equal to 0 or 1 as an argument, and returns 0 or 1. When a process p_i invokes $propose(v)$, we say that p_i proposes value v. When p_i has returned value v' from $propose(v)$, we say that p_i decides value v' (notice that v' does not have to be equal to v). A binary consensus object satisfies the following properties:

Agreement No two processes decide different values.

Validity The value decided is one of the values proposed.

A *write-once register* is a shared object with the following sequential specification. Assume x is initially equal to $\perp$ and v is always different than $\perp$. You may assume that calls to $write()$ are atomic.

```
upon write(v)
  if x = ⊥ then x := v
  return ok

upon read
  return x
```

Your tasks are:

1. To implement a binary consensus object using any number of write-once registers;
2. Explain why your algorithms satisfies the **Agreement** and **Validity** properties.

Problem 2. Consider the *binary consensus* problem from the previous exercise. Recall that this abstraction satisfies the following properties:

Agreement No two processes decide different values.

Validity The value decided is one of the values proposed.

Recall that processes might in fact *crash*, i.e. stop taking steps, at any point during the execution of an algorithm. We add an extra condition on progress:

Termination Every process that does not crash will eventually decide.

Assume the following theorem is true:

Theorem 1 (FLP) *Binary consensus is impossible among N processes, if one of them might fail by crashing, in an asynchronous system that disposes of binary SRSW safe registers*[1].

[1] In fact, this is one of the major results in distributed computing, first stated by Fisher, Lynch and Patterson, in their paper "Impossibility of Distributed Consensus with One Faulty Process" in 1985. For details, see the presentation in the Encyclopedia of Algorithms (available on Google Books), or the original paper, which is quite readable.

Using what you already know about registers, prove the following result. (Hint: you may use existing results given in the 1st and 2nd week lectures.)

Theorem 2 (Students' FLP) *Binary consensus is impossible among N processes, if one of them might fail by crashing, in an asynchronous system that disposes of MRMW atomic registers.*

Problem 3. Assuming the results in Problem 2, prove or disprove the following statement:

Theorem 3 (Write-Once Registers) *There is no implementation of a write-once register from MWMR atomic registers.*

Problem 4. In the lecture, you have seen that consensus can be solved among two processes given shared registers and Test&Set objects (or Fetch&Inc objects). Consider a special shared object X that allows two atomic operations, Test&Set, Fetch&Add2, and maintains an integer x initialized to 0. Its sequential specification is as follows.

```
Test&Set() {
    ret := x;
    if(x = 0) x := 1;
    return (ret);
}
Fetch&Add2(){
    ret := x;
    x := x + 2;
    return (ret);
}
```

We say that a shared object X has consensus number k if:

- there exists an algorithm that implements wait-free consensus among k processes using any number of instances of X and atomic *MRMW* registers.
- no such algorithm exists for $k+1$ processes.

Does X have consensus number ∞? If yes, provide an algorithm that solves binary consensus among an arbitrary number n of processes, using X (and arbitrarily many shared registers); justify that your algorithm is correct. If not, show that for a specific number k, no algorithm using X (and arbitrarily many shared registers) solves binary consensus among k processes.