Concurrent Algorithms October 31, 2017

Solutions to Exercise 5

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 $\perp$.

```
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. To implement a binary consensus object using one or more (shared) queue objects in a system of 2 processes.
3. Explain why your algorithms satisfy the **Agreement** and **Validity** properties.

Solution

The solution can be found on the Exercise 5.1 solution slides on the course website.

Problem 2. Consider the *binary consensus* problem. 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.

Let's assume 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].

Using what you know about registers, prove the following result:

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.*

Solution

Assume for the sake of contradiction that there exists an algorithm $\mathcal{A}$ that solves consensus in an asynchronous system using MRMW atomic registers. We prove that in this case there exists an algorithm $\mathcal{A}'$ that solves consensus in an asyncronous system using SRSW safe registers, which contradicts the FLP theorem.

Let us write down the code of algorithm $\mathcal{A}$. The algorithm makes use of a (possibly infinite) number of MRMW atomic registers $R_1, R_2, \ldots$. We know from the course that there exists an implementation of MRMW atomic registers using (an infinite number of) SRSW safe registers. Therefore, for each call of R_i.read or R_i.write that the algorithm $\mathcal{A}$ makes, we replace the call with the implementation of R_i using only SRSW safe registers. We call the resulting algorithm code $\mathcal{A}'$.

To conclude the proof, we notice that algorithm $\mathcal{A}'$ implements consensus using only SRSW registers. The correctness of $\mathcal{A}'$ follows from the (assumed) correctness of $\mathcal{A}$ and the correctness of the register implementation, which has been shown in class.

[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.

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.*

Solution

Follows from 1 and 2.