Concurrent Algorithms: Exercise 2

October 19, 2009

Problem 1

1. Devise an algorithm that implements an atomic M-valued SWMR register using (any number of) atomic binary SWMR registers.
2. Prove that your algorithm is correct.
3. Explain whether your algorithm remains correct (i.e., implements an atomic register) if you change the binary base registers from atomic to regular.

Problem 2

Consider the *binary consensus* problem from the previous exercise sheet. 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.*

[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.

Solution for Problem 1

```
operation write(v)
    r[v].write(1);
    for i ← v − 1 downto 0 do r[i].write(0);
    return ok;
end

operation read
    v ← 0;
    while r[v].read = 0 do v ← v + 1;
    for i ← v − 1 downto 0 do
        if r[i].read = 1 then v ← i;
    end
end
```

Sketch of Proof

A useful tool for checking correctness of atomic SWMR registers implementations is the following Lemma. You can use it without proof in the following problem sets and in the exam.

Definition 3 (Read-Inversions) *A* read inversion *is a particular execution fragment in an implementation of a SWMR register, where the following occurs: during a write* W1 *that writes v_2 instead of v_1 ($v_1 \neq v_2$), there are two separate reads* R1 *and* R2 *with the property that*

- R1 *finishes* before R2 *starts;*
- R1 *returns v_2, the new value written;*
- R2 *returns v_1, the old value.*

Lemma 4 *An implementation of atomic registers is correct if the implementation is regular, and there are no* read- inversions.

It is relatively easy to check that the solution above satisfies atomicity using Lemma 1. You should try it as an exercise. Otherwise, you can take a look at the complete proof, which can be found in the third part of the lecture notes from the course website:

`https://lpd.epfl.ch/site/_media/education/chap3.ps`

Solution for Problem 2

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.