A Solution to Exercise 5

Concurrent Algorithms 2013

LPD, EPFL

Exercises 1.a and 1.b

The solution was given in the lecture on the limitations of registers.

Exercise 1.c

The consensus number of the queue object is 2

1. One can implement consensus between 2 processes using only queues and atomic registers (done in class).

2. **One cannot implement consensus between 3 processes using only queues and atomic registers.**

Exercise 1.c

We want to prove:

Binary consensus between 3 processes is impossible using only queues and registers.

Idea: **proof by contradiction**; we assume we have a consensus algorithm between three processes p1, p2 and p3, and show this leads to contradictions.

Essentially, we will show that there is at least one bivalent initial configuration, and that for any bivalent configuration there is a schedule that leads to another bivalent configuration, hence contradicting the hypothesis.

Exercise 1.c

(Lemma 1): There exists an initial bivalent configuration in a system of 3 processes using queues and atomic registers.

Proof (as in the lecture):

We show the initial configuration C(0,1,1) is bivalent:

Consider C(0,0,0) and p2 and p3 not taking any steps: p1 decides 0; p1 cannot distinguish C(0,0,0) from C(0,1,1) and can hence decides 0 starting from C(0,1,1); similarly, if we consider C(1,1,1) and p1 and p3 not taking any step, p2 eventually decides 1; p2 cannot distinguish C(1,1,1) from C(0,1,1) and can hence decides 1 starting from C(0,1,1). Hence the bivalency.

Exercise 1.c

(Lemma 2): If C_i and C_j are indistinguishable to process p_k, then they must have the same valency.

! C_i, C_j – indistinguishable to process p_k: the states of the shared objects are the same in C_i and C_j, and the state of p_k is the same

Proof:

Assume p_k performs schedule *s* from C_i and decides *v*; if we apply *s* to C_j, the state of the shared objects and p_k in $s(C_j)$ and $s(C_i)$ are the same. Therefore p_k should decide the same value in $s(C_j)$.

Exercise 1.c

(Lemma 3): For any bivalent configuration there is an arbitrary long schedule which leads to another bivalent configuration.

Proof: Assume there is no such schedule, show contradiction.

C_{init} – initial bivalent configuration;

C – bivalent configuration s.t. for any *s*, *s*(C) is univalent;

e_1, e_2, e_3 – single steps performed by processes p_1, p_2, p_3 respectively

$e_1(C)$, $e_2(C)$, $e_3(C)$ – univalent, but not all have the same valency (since C - bivalent); assume $e_1(C)$ is 0-valent, $e_2(C)$ is 1-valent;

Exercise 1.c

Proof (cont'd):

We analyze e1 and e2. Assume they access the same object (otherwise e1(e2(C)) = e2(e1(C))). If they access a register, use the arguments in the FLP proof to reach the conclusion. Therefore, we assume they access a queue.

Case 1: e1, e2 – both dequeues:

e1(e2(C)), e2(e1(C)) – indistinguishable to p3 => same valency (per Lemma 2); contradiction.

Case 2: e1 – dequeue, e2 – enqueue;

if Q – not empty in C, then e1(e2(C)) and e2(e1(C)) – indistinguishable to p3 => same valency; contradiction.

if Q – empty in C, e2(C) and e2(e1(C)) – indistinguishable to p3 => same valency (Lemma 2); contradiction.

Exercise 1.c

Proof (cont'd):

Case 3: e1, e2 – enqueues;

Let a,b – the values enqueued by p1 and p2 respectively

The processes must run until they dequeue a or b, otherwise they cannot distinguish between e1(C) and e2(C).

Execution s1(C):	**Execution s2(C):**
1. p1 and p2 enqueue a and b in that order 2. p1 runs until it dequeues a 3. p2 runs until it dequeues b	1. p2 and p1 enqueue b and a in that order 2. p1 runs until it dequeues b 3. p2 runs until it dequeues a
0 – valent (since e1(C) – 0-valent)	**1 – valent** (since e2(C) – 1-valent)

p1's executions – identical until it dequeues a or b; no modifications afterwards;
p2's executions – identical until it dequeues a or b; no modifications afterwards;

=> s1(C) and s2(C) identical to p3, hence same valency (Lemma 2) => contradiction.

Exercise 1.c

Assume there is an algorithm implementing consensus between 3 processes using queues and atomic registers.

(Lemma 1): There exists an initial bivalent configuration in a system of 3 processes using queues and atomic registers.

(Lemma 2): If C_i and C_j are indistinguishable to process p_k, then they must have the same valency.

(Lemma 3): For any bivalent configuration there is an arbitrary long schedule which leads to another bivalent configuration.

Lemmas 1, 3 => contradiction with the hypothesis

There is no wait-free implementation of a consensus object using queues and registers in a systems of 3 processes.

Exercise 1.c

Complete solution on the website, also see [*Herlihy, M. P.* **Wait-free synchronization**. ACM Transactions on Programming Languages and Systems, 13(1):124—149, January 1991].

Exercise 2

2 process consensus using only uninitialized queues and atomic registers

- Shared objects: a queue Q, atomic register R, array of atomic registers Input[2]
- Q - initially empty;
- P1 inserts in the queue Q, P2 writes to R;

Exercise 2

Process 1:

```
propose₁(val₁):
    Input[1] := val₁;
    Q.enqueue(lose);
    if R.read() = 1 then
        if Q.dequeue()= empty then return Input[1];
        else return Input[2];
    else return Input[1];
```

Process 2:

```
propose₂(val₂):
    Input[2] := val₂;
    R := 1;
    if Q.dequeue() = empty then return Input[2];
    else return Input[1];
```