STiDC'07: Exercise 2

October 8, 2007 (updated on October 22, 2007)

1 Problem 1

In Exercise 1, we were implementing a binary consensus object from a queue initialized to $\langle$ *winner*, *loser* $\rangle$ and two atomic registers, in a system of 2 processes. Write an algorithm that implements binary consensus for 2 processes using (any number of) queue objects that are *initially empty* and atomic registers.

2 Problem 2

Assume we have a shared object Q that implements, among others, an operation $init(s)$ that atomically changes the state of Q to s. Let A be an algorithm that implements n-process consensus using object Q initialized to some state $q \neq \perp$. Find an algorithm B that implements n-process consensus using algorithm A, a number of instances of object Q initialized to $\perp$, and atomic registers, or prove that such an algorithm does not exist.

3 Solutions

First, let us recall the binary consensus algorithm for 2 processes using a queue Q initialized to $\langle$ *"winner"*, *"loser"* $\rangle$ and 2 atomic registers $R[1,2]$:

procedure $cons_i(Q, R, val_i)$
- $R[i] \leftarrow val_i$
- $q_i \leftarrow Q.deq()$
- **if** $q_i =$ *"winner"* **then return** val_i
- **else return** $R[3-i]$

Now, we will implement a binary consensus algorithm for 2 processes using 2 queues, Q_1 and Q_2, that are initially empty and 6 atomic registers, $R_{1,2}[1,2]$ and $ready_{1,2}$ (initialized to *false*). The basic idea is the following: each process p_i ($i = 1,2$) first initializes queue Q_i to $\langle$ *"winner"*, *"loser"* $\rangle$ and sets register $ready_i$ to *true*. Once queue Q_i is initialized, each process can run the above consensus algorithm (procedure *cons*) using Q_i and registers $R_i[1,2]$. Process p_i, after initializing queue Q_i, runs $cons_i$ for each queue that is already initialized (i.e., only Q_i, or both Q_i and Q_{3-i}) in an order common for both processes. An important thing to note is that if p_i decides some value v in the first consensus (using Q_1), then p_i proposes v to the other consensus (using Q_2). The exact algorithm is the following:

upon $propose_i(val_i)$
- $Q_i.enq($*"winner"*$)$
- $Q_i.enq($*"loser"*$)$
- $ready_i \leftarrow true$
- **for** $k \leftarrow 1,2$ **do**
  - **if** $ready_k$ **then** $val_i \leftarrow cons_i(Q_k, R_k, val_i)$
- **return** val_i

It is straightforward to generalize the above algorithm and thus solve Problem 2. The algorithm is the following:

upon $propose_i(val_i)$
  $Q_i.init(q)$
  $ready_i \leftarrow true$
  for $k \leftarrow 1, \ldots, n$ **do**
    if $ready_k$ **then** $val_i \leftarrow cons_i(Q_k, val_i)$
  return val_i

Where: $Q_{1,\ldots,n}$ are instances of shared object Q initialized to $\perp$ and $cons_i(Q_k, val_i)$ is an n-consensus algorithm (at process p_i) that uses object Q_k initialized to state q.