Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own.

Notice 2: Unless your hand writing is extremely clear, you should type (Word, or LaTeX, etc.) and print your homework.

Notice 3: Put your name and id number on each page of the solution.

1. The roman senate strikes again, this time they want to reach agreement (consensus) under the same conditions given in the previous homework: They want to agree on the binary input of one of them.

The senators gather in a ring around the roman forum, and the announcer declares the method (distributed algorithm) by which they will reach consensus. Senators then converse, each with his two neighbors on the ring, according to the announced method (distributed algorithm).

However, each senator might prefer that either 0 or 1 be decided, and thus may cheat in an attempt to increase the probability of its preferred value being decided. You have to devise a consensus algorithm that the announcer will declare to the senators, such that, no senator will gain anything from deviating (cheating) from your algorithm.

Your algorithm should satisfy both the agreement and the validity conditions. The cheating is constrained as follows:

(a) If the algorithm fails to reach agreement, the senators are sent to prison for the rest of their life. Meaning, senators will not cheat if the cheating will definitely cause the algorithm failure. However, senators do take chances, if there is some positive probability to succeed.

(b) A senator cheats only if the cheating increases the probability of its preferred value being decided.

(c) Cheating senators assume they are the only cheaters.

Further make the following assumptions: (i) each senator has a unique name (known only to itself), (ii) each knows $n$, the total number of senators (iii) the senators converse in synchronous rounds (i.e., assume a synchronous ring), and (iv) assume all senators start together at the same round.

For any algorithm that you provide give its message and time complexities.

(a) Provide an algorithm when each senator prefers its own binary input.
(b) Provide an algorithm when each senator prefers a certain value (0 or 1) not necessarily its own.

(c) Provide an algorithm when a subset of the senators may cooperate in cheating, to increase the probability of a certain value. Each subset assumes that it is the only one to cheat. What is the largest size sub-set under which your algorithm works correctly? It should be the maximum size for which there is a solution. Notice, members of any subset may communicate only through the ring.

(d) Repeat the above for multi-value consensus. That is each agent input is from domain $V = \{1, \ldots, n\}$.

(e) How can you eliminate the assumption that all start together? (should work for all the above sub-questions).

(f) Repeat the above questions for an synchronous complete network.

2. Consider the following two tasks:

The $g$-tight group renaming task: In a tight group renaming task with group size $g$, $n$ processors with id’s from a large domain $\{1, 2, \ldots, N\}$ are partitioned into $m$ groups with id’s from a large domain $\{1, 2, \ldots, M\}$, with at most $g$ processors per group. A tight group renaming task renames groups from the domain $1..M$ to $1..l$ for $l << M$, where all processors with the same initial group ID are renamed to the same new group ID, and no two different initial group id’s are renamed to the same new group ID.

The $n$-Safe-consensus problem: In this task $n$ processors with id’s $1..n$ each receives a private input value, and outputs a value such that:

- Wait-Free: Each processor finishes executing within a finite number of its own steps.
- Agreement: All processors output the same output value.
- Weak-Validity: If the output of a processor occurs before the invocation of any other processor then the output is that processor’s proposed input value.

Notice this is not a serial definition. If no processor initially accesses (runs) the Safe-consensus task in exclusion then processors may agree on any value (even one that is not in the input domain). Notice, a similar task but in which the agreement condition and the validity condition are slightly changed is read/write implementable (see question 3.b).

(a) Show how to implement $k$-Safe-consensus task in a system equipped with a $k$-tight group renaming task (for groups of size $k$) that may be accessed $k + 1$ times.

(b) Prove that the above is impossible if the tight group renaming task may be accessed only $k$ times.

2
(c) Show that the power (consensus wise) of \(k\)-Safe-consensus and \(k\)-consensus is the same, i.e., that with enough copies of one we can implement the other.

3. In exclusion-terminating \(\text{with default}\) consensus task \(T\) is invoked by calling \(\text{propose}(\text{input})\), which returns an output \(o\) that can be either a value or \text{default}, such that:

**Agreement:** If an invocation of \(\text{propose}(\text{input})\) returns \(o = v, v \neq \text{default}\) then any other invocation of \(\text{propose}(\text{input})\) returns \(o = v\) or \(o = \text{default}\).

**Validity:** If an invocation of \(\text{propose}(\text{input})\) returns \(o = v\) which is not \text{default} then there is an invocation of \(\text{propose}(v)\) by some process, which does not return \text{default}.

**Exclusion-Termination with default:** If during the entire invocation of \(\text{propose}(\text{input})\) by some process \(p\) no other process took any step, then \(p\) returns \(o \neq \text{default}\) (the other process may be after the invocation and before the response but not taking any steps, i.e., not accessing any of the primitive objects). Any other invocation of \(\text{propose}(\text{input})\) eventually returns a value \(o\) which is either \text{default} or some value \(v\).

(a) (try in 1 page, but at most 2 pages) Show that there is no implementation of such a task using read/write registers. Hint: show this impossibility by a reduction to wait-free consensus for two processes.

(b) (try in 1 page, but at most 2 pages) In this section we change the following condition: **Exclusion-Termination with default** condition is changed as follows: If process \(p\) completes its entire invocation of \(\text{propose}(\text{input}_p)\) before any other process takes any step then \(p\) returns \(o = \text{input}_p\). Any other invocation of \(\text{propose}(\text{input})\) eventually returns a value \(o\) which is either \text{default} or \(v\). Is this task read/write implementable?