Computer Science **Homework 1 Answers**

Behdad Esfahbod **Student # 993505827**

---

1. Consider that both p and q start with 0, and all messages are lost, then by *Termination* and *Strong Validity* rules, we conclude that they both decide 0. Now consider the case that both start with 1, and again all messages are lost, then they have to both decide 1.

   Now, if p starts with 0, and q with 1, and all messages are lost, from the viewpoint of p, it is the same scenario as the first consideration above, so it would decide 0, and for q, it is the same as second case above, so it would decide 1, which contradicts the *Uniform Agreement* rule.

2. It can be solved by the following simple protocol:

   Round 1: Each process sends its input value to the other process.

   Round 2: For every process:

   (a) If received something from the other process, then deciede the logical AND of the received number and your own input value.

   (b) Otherwise do not decide.

   proof If any of the processes decides, it has done it based on the input value of both of them, and the rule to decide satisfies both *Uniform Agreement* and *Validity*. For *Weak Termination*, we observe that if no failures happen, then both of them would decide in the second run.

3. We give an impossibility proof. The idea is that the same impossiblity proof that was presented in the class works here: Lets A be an algorithm which guarantees to solve the problem with *Unanimous Termination*. Then if no failures happen, they both should decide. Now we start removing the last message, and since one of them would not feel different, it would still decide, so the other one should decide as before, but with one fewer message transfered. Repeating the same technique, we would see that they should decide even if all messages are lost. Now we can easily show that if p starts with 0 and q with 1, and all messages are lost, then p should decide 0, and q should decide 1, which is a contradiction with the fact that algorithm A satisfies *Agreement*. So no such algorithm exists.

4. (a) The following algorithm solves the problem:
   In each round i do:

   i. If not decided yet, send a *request* message to the other process.

   ii. If received a *request* message from the other process, send your input value to the other process.

   iii. If received from the other process its input value, decide and logical AND of your own input value and the received value.

proof *Validity* and *Agreement* are obviously satisfied, as they both decide the logical AND of their input values, which would be the same for both os them, and satisfies *Validity*.

For *Termination*, what happens is that each process repeats sending the other one *request* messages. As the links are fair, this *request* message would eventually arrive. Moreover, we can assume that the message is arrived as many times as we want, as the sender can continue sending it, and it should arrive again by the hypothesis. So, the other process, receives the *request* message, and replies with its input value. This scenario repeats, until the first process receives the input value of the other process. And it would eventually happen. So each one would terminate. And after both have decided, they would become quiet.

(b) In that impossiblity proof, we assumed that we can lose all messages after a certain time, and the processes have to decide the same way they were doing. Now with fair links, we cannot do that. For example in the algorithm above, if we break the link, they would send *request* messages infinitly many times, and would never decide.

(c) We go to prove that no such *halting* algorithm exists. The proof is pretty like the one presented in the class, with the difference that we cut the links up to a certain time.

Suppose there exists and algorithm solving the problem. So considering a scenario that both p and q start with 0, and they finally decide 0 and halt. Now assume that the last message is sent by q to p, and q halts in time t. So now we decide that this last message and all messages after that until time t are going to be lost. Now, q do not feel any difference, and halts as before, as p cannot send any messages to q before time t. So p should decide 0 and halt too, and we have one fewer message. Continuing the argument, we can show that with all messages lost until a certain time t, the should both decide 0 and halt before t.

Not again, doing the same discussion with 1, shows that they should decide 1, with all messages lost before t. Combining the too, p starting with 0, and q with 1, means that p should decide 0, and q should decide 1, which is a contradiction. So no such algorithm exists.