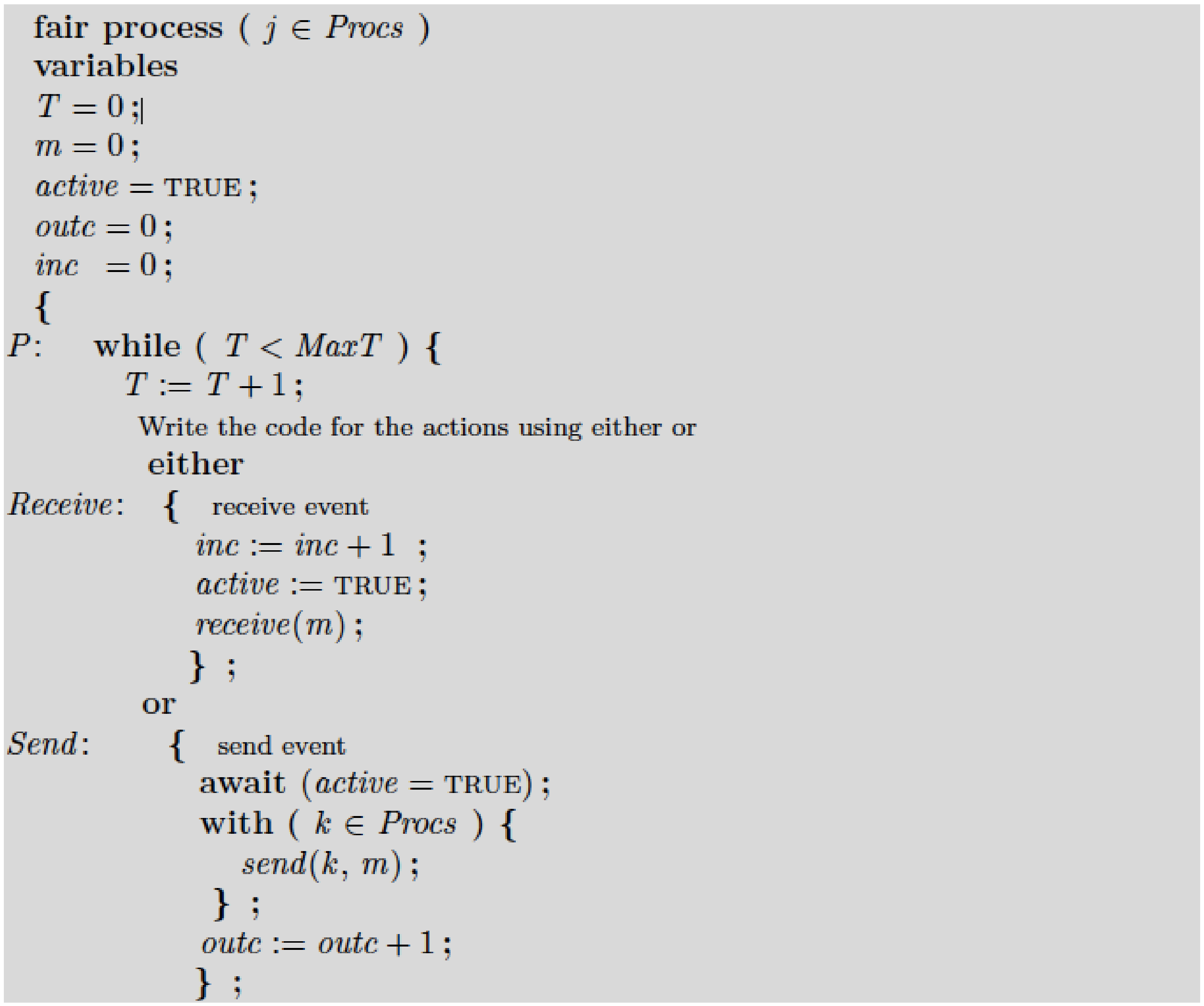
CSE 4/586 Project: Termination detection

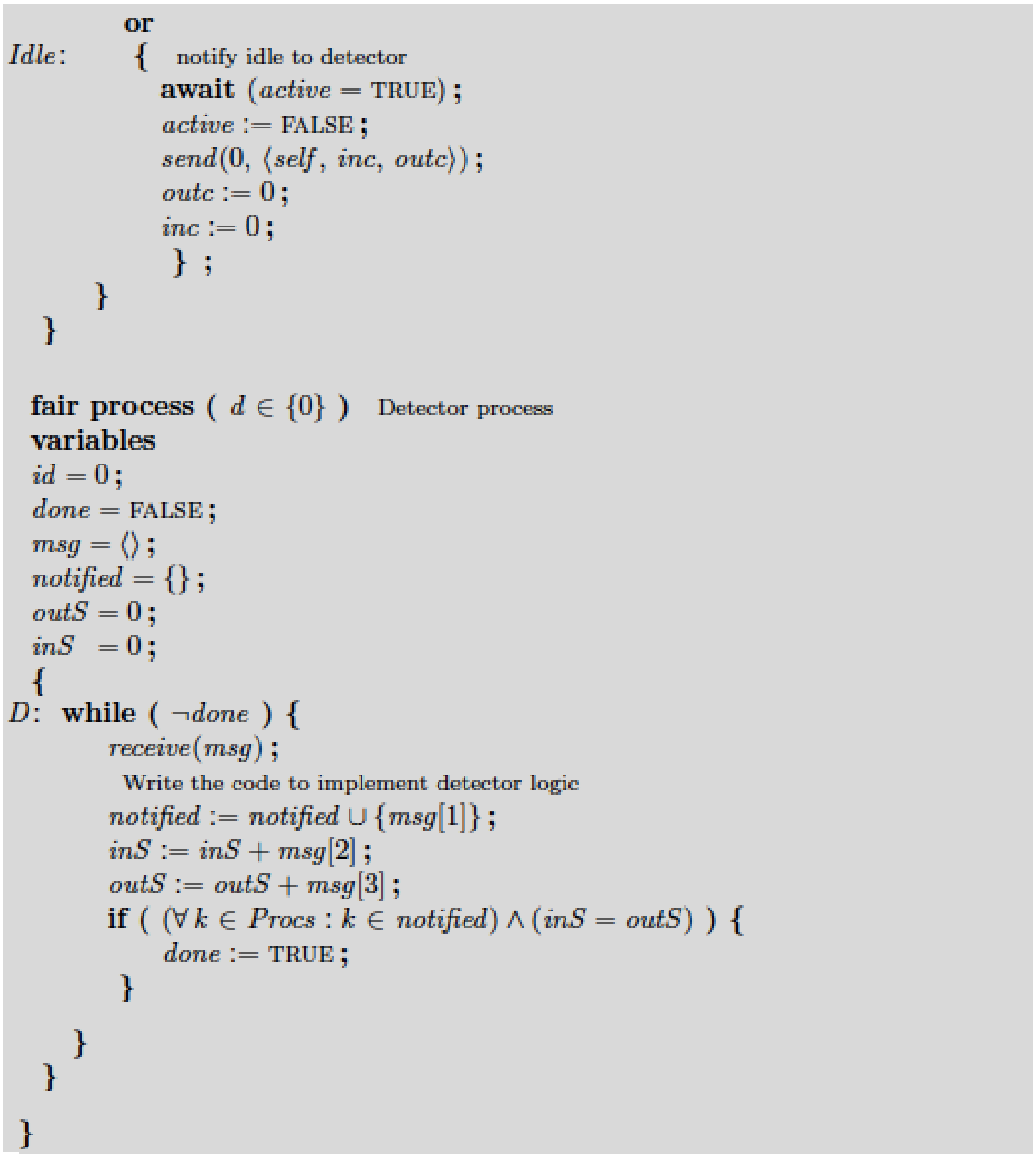
GROUP MEMBERS
Upasana Ghosh (UB Person #**50317396**)
Sriparna Chakraborty (UB Person #**50314303**)

Part 1: Termination Detection Algorithm (Each process keeps track of the total messages sent and the total messages received)

Below is the implementation of the first part:



```
  fair process ( $j \in Procs$ )
  variables
  $T = 0$;
  $m = 0$;
  $active = \text{TRUE}$;
  $outc = 0$;
  $inc = 0$;
  {
$P$:    while ( $T < MaxT$ ) {
          $T := T + 1$;
           Write the code for the actions using either or
           either
$Receive$:  {  receive event
              $inc := inc + 1$ ;
              $active := \text{TRUE}$;
              $receive(m)$;
             } ;
           or
$Send$:     {  send event
              await ($active = \text{TRUE}$);
              with ( $k \in Procs$ ) {
                 $send(k, m)$;
               } ;
              $outc := outc + 1$;
             } ;
```



```
           or
$Idle$:       {   notify idle to detector
              await $(active = \mathrm{TRUE})$;
              $active := \mathrm{FALSE}$;
              $send(0, \langle self, inc, outc \rangle)$;
              $outc := 0$;
              $inc := 0$;
               } ;
         }
   }

  fair process ( $d \in \{0\}$ )   Detector process
  variables
  $id = 0$;
  $done = \mathrm{FALSE}$;
  $msg = \langle\rangle$;
  $notified = \{\}$;
  $outS = 0$;
  $inS \ \ = 0$;
  {
$D$:  while ( $\neg done$ ) {
          $receive(msg)$;
           Write the code to implement detector logic
          $notified := notified \cup \{msg[1]\}$;
          $inS := inS + msg[2]$;
          $outS := outS + msg[3]$;
          if ( $(\forall k \in Procs : k \in notified) \wedge (inS = outS)$ ) {
              $done := \mathrm{TRUE}$;
           }

       }
   }

 }
```

The above termination detection algorithm violates the safety property which is defined as follows:

```
Safety == (done[0] => ((\A k \in Procs: active[k] = FALSE)
                /\ (\A f \in Procs: chan[f] = <<>>)))
```

The model checker comes up with the following counter-example to prove the flaw in the algorithm while running for N=3 and MaxT=3:

```
State (num=18)
/\  T = <<2, 3, 2>>
/\  active = <<FALSE, TRUE, FALSE>>
/\  chan = (0 :> <<>> @@ 1 :> <<>> @@ 2 :> <<>> @@ 3 :> <<>>)
/\  done = (0 :> TRUE)
/\  id = (0 :> 0)
/\  inS = (0 :> 1)
/\  inc = <<0, 1, 0>>
/\  m = <<0, 0, 0>>
/\  msg = (0 :> <<3, 0, 1>>)
/\  notified = (0 :> {1, 2, 3})
/\  outS = (0 :> 1)
/\  outc = <<0, 1, 0>>
/\  pc = (0 :> "D" @@ 1 :> "P" @@ 2 :> "P" @@ 3 :> "P")
```

The following steps show our observation and clarification about what went wrong:

- There are three processes (N=3) and the notified set in the example above consists of {1,2,3}, which implies that all the processes have notified the detector that they are in an idle state and we also see that inS = outS (=1).
- According to the Termination detection condition, at this point, all the computation in the system can be considered as terminated and the done flag for the detector should be set to True. In the example, we see that the done = TRUE.
- But, to verify that all the computations in the system have terminated, each of the processes in the system should be in an idle state, i.e., active = FALSE for all processes. But we observe that the active status for process 2 is TRUE, i.e., process 2 is still active which means that all the computations have not yet terminated. Hence violating the safety property for this program.
- This happens because we assume in the algorithm that just by comparing the total messages received with the total messages sent, the detector can predict whether the system has reached termination. But, this information is not enough for the detector process to come to a decision.
- This is because we do not take into consideration the state of the channels over which the communications happen.
- In the above example, it could have happened that process 2 received a message that was still in transit, after informing the detector about its idle state. The detector finding that the inS=outS and seeing that all the processes have notified it about their idle state, turns the done flag on unaware of the change in state at process 2.
- Hence violating the safety property.