```
(*
Pratik Kubal 50290804 Anirudh CR 50290168 *)
---------------------------- MODULE BenOr ----------------------------
EXTENDS Integers, TLC, Sequences, FiniteSets
CONSTANT N, F, MAXROUND, INPUT
 \* N Nodes, F Failures
ASSUME N ∈ Nat ∧ F ∈ Nat ∧ F ≤ N
 \* Assuming INPUT is a valid Sequence in the form ⟨x, .., N⟩
Procs ≜ 1 .. N
    --fair algorithm BenOr
    {
        variable p1Msg = {}, p2Msg = {};

        define
        {
            ExtractValSet(S)  ≜ {m.value : m ∈ S}
            CollectP1Msgs(r) ≜ {m ∈ p1Msg : (m.round = r)}
            CollectP2Msgs(r) ≜ {m ∈ p2Msg : (m.round = r)}
            ValueMsg(r, v) ≜  {m ∈ p1Msg : (m.round = r) ∧ (m.value = v)}
            ValueMsgP2(r, v) ≜  {m ∈ p2Msg : (m.round = r) ∧ (m.value = v)}

         }

        macro SendP1( r, i )
        {
             \* Sends initial value i
             print(i);
            p1Msg := p1Msg ∪ {[nodeid ↦ self, round ↦ r, value ↦ i]};

         }

        macro RvcP1( r )
        {
             \* The below statement gives A which is first N − F messages received
            await (Cardinality(CollectP1Msgs(r)) ≥ N − F);

             print("inside recieve");
            if ( (Cardinality(ValueMsg(r, 1)) * 2 > N) ) {
            p2v := 1;
             print("1");
             }
            else if ( (Cardinality(ValueMsg(r, 0)) * 2 > N) ) {
              print("0");
            p2v := 0;
             }
            else {
             print("− 1");
```

```
    p2v := -1;
     }
 }

 macro SendP2( r, i )
{
      \* Sends initial value i
    p2Msg := p2Msg ∪ {[nodeid ↦ self, round ↦ r, value ↦ i]}
 }

macro RvcP2( r )
{
      \* The below statement gives A which is first N - F messages received
    await (Cardinality(CollectP2Msgs(r)) ≥ N - F);

     print("inside recieve");
    if ( Cardinality(ValueMsgP2(r, 1)) ≥ F + 1 ) {
    decided := 1;
     }
    else if ( Cardinality(ValueMsgP2(r, 0)) ≥ F + 1 ) {
    decided := 0;
     } ;
    if ( (Cardinality(ValueMsgP2(r, 0)) > 0) ) {
    p1v := 0;
     }
    else if ( (Cardinality(ValueMsgP2(r, 1)) > 0) ) {
    p1v := 1;
     }
    else if ( (Cardinality(ValueMsgP2(r, -1)) ≥ 1) ) {
    with ( v ∈ {0, 1} ) {
        p1v := v;
     } ;
     } ;
 }

fair process ( p ∈ Procs )
variable r = 1, p1v = INPUT[self], p2v = -1, decided = -1;
{
    entry: while ( r < MAXROUND )
    {
          \* SendP1 → macro which will post the value of that node to the message board as p1v
        P1S: SendP1(r, p1v);
          \* RvcP1 → Get n-f values with p1v
        P1R: RvcP1(r);
          \* SendP2 → Macro which sends the b value of the node to the message board as p2v
        P2S: SendP2(r, p2v);
```

$\backslash * \; RvcP2 \rightarrow$ Get n-f values with $p2v$

```
                    P2R: {
                    RvcP2(r) ;
                    r := r + 1 ;
                    } ;
                    } ;
                } ;
            }
    }
```

BEGIN TRANSLATION

VARIABLES $p1Msg,\ p2Msg,\ pc$

define statement

$ExtractValSet(S) \triangleq \{m.value : m \in S\}$

$CollectP1Msgs(r) \triangleq \{m \in p1Msg : (m.round = r)\}$

$CollectP2Msgs(r) \triangleq \{m \in p2Msg : (m.round = r)\}$

$ValueMsg(r,\ v) \triangleq \{m \in p1Msg : (m.round = r) \land (m.value = v)\}$

$ValueMsgP2(r,\ v) \triangleq \{m \in p2Msg : (m.round = r) \land (m.value = v)\}$

VARIABLES $r,\ p1v,\ p2v,\ decided$

$vars \triangleq \langle p1Msg,\ p2Msg,\ pc,\ r,\ p1v,\ p2v,\ decided\rangle$

$ProcSet \triangleq (Procs)$

$Init \triangleq$ Global variables

$\land\ p1Msg = \{\}$

$\land\ p2Msg = \{\}$

Process p

$\land\ r = [self \in Procs \mapsto 1]$

$\land\ p1v = [self \in Procs \mapsto INPUT[self]]$

$\land\ p2v = [self \in Procs \mapsto -1]$

$\land\ decided = [self \in Procs \mapsto -1]$

$\land\ pc = [self \in ProcSet \mapsto \text{“entry”}]$

$entry(self) \triangleq \land\ pc[self] = \text{“entry”}$

$\quad \land$ IF $r[self] < MAXROUND$

$\qquad$ THEN $\land\ pc' = [pc$ EXCEPT $![self] = \text{“P1S”}]$

$\qquad$ ELSE $\land\ pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$

$\quad \land$ UNCHANGED $\langle p1Msg,\ p2Msg,\ r,\ p1v,\ p2v,\ decided\rangle$

$P1S(self) \triangleq \land\ pc[self] = \text{“P1S”}$

$\quad \land\ p1Msg' = (p1Msg \cup \{[nodeid \mapsto self,\ round \mapsto r[self],\ value \mapsto p1v[self]]\})$

$\quad \land\ pc' = [pc$ EXCEPT $![self] = \text{“P1R”}]$

$\quad \land$ UNCHANGED $\langle p2Msg,\ r,\ p1v,\ p2v,\ decided\rangle$

$P1R(self) \triangleq \land\ pc[self] = \text{“P1R”}$

$$
\begin{array}{l}
\land\ (Cardinality(CollectP1Msgs(r[self])) \geq N - F) \\
\land\ \text{IF } (Cardinality(ValueMsg(r[self], 1)) * 2 > N) \\
\quad\quad \text{THEN } \land\ p2v' = [p2v \text{ EXCEPT } ![self] = 1] \\
\quad\quad \text{ELSE } \land\ \text{IF } (Cardinality(ValueMsg(r[self], 0)) * 2 > N) \\
\quad\quad\quad\quad\quad\quad \text{THEN } \land\ p2v' = [p2v \text{ EXCEPT } ![self] = 0] \\
\quad\quad\quad\quad\quad\quad \text{ELSE } \land\ p2v' = [p2v \text{ EXCEPT } ![self] = -1] \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“P2S”}] \\
\land\ \text{UNCHANGED } \langle p1Msg, p2Msg, r, p1v, decided \rangle
\end{array}
$$

$$
\begin{array}{lcl}
P2S(self) & \triangleq & \land\ pc[self] = \text{“P2S”} \\
 & & \land\ p2Msg' = (p2Msg \cup \{[nodeid \mapsto self, round \mapsto r[self], value \mapsto p2v[self]]\}) \\
 & & \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“P2R”}] \\
 & & \land\ \text{UNCHANGED } \langle p1Msg, r, p1v, p2v, decided \rangle
\end{array}
$$

$$
\begin{array}{lcl}
P2R(self) & \triangleq & \land\ pc[self] = \text{“P2R”} \\
 & & \land\ (Cardinality(CollectP2Msgs(r[self])) \geq N - F) \\
 & & \land\ \text{IF } Cardinality(ValueMsgP2(r[self], 1)) \geq F + 1 \\
 & & \quad\quad \text{THEN } \land\ decided' = [decided \text{ EXCEPT } ![self] = 1] \\
 & & \quad\quad \text{ELSE } \land\ \text{IF } Cardinality(ValueMsgP2(r[self], 0)) \geq F + 1 \\
 & & \quad\quad\quad\quad\quad\quad \text{THEN } \land\ decided' = [decided \text{ EXCEPT } ![self] = 0] \\
 & & \quad\quad\quad\quad\quad\quad \text{ELSE } \land\ \text{TRUE} \\
 & & \quad\quad\quad\quad\quad\quad\quad\quad\quad\ \land\ \text{UNCHANGED } decided \\
 & & \land\ \text{IF } (Cardinality(ValueMsgP2(r[self], 0)) > 0) \\
 & & \quad\quad \text{THEN } \land\ p1v' = [p1v \text{ EXCEPT } ![self] = 0] \\
 & & \quad\quad \text{ELSE } \land\ \text{IF } (Cardinality(ValueMsgP2(r[self], 1)) > 0) \\
 & & \quad\quad\quad\quad\quad\quad \text{THEN } \land\ p1v' = [p1v \text{ EXCEPT } ![self] = 1] \\
 & & \quad\quad\quad\quad\quad\quad \text{ELSE } \land\ \text{IF } (Cardinality(ValueMsgP2(r[self], -1)) \geq 1) \\
 & & \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{THEN } \land\ \exists\, v \in \{0, 1\} : \\
 & & \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad p1v' = [p1v \text{ EXCEPT } ![self] = v] \\
 & & \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{ELSE } \land\ \text{TRUE} \\
 & & \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \land\ p1v' = p1v \\
 & & \land\ r' = [r \text{ EXCEPT } ![self] = r[self] + 1] \\
 & & \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“entry”}] \\
 & & \land\ \text{UNCHANGED } \langle p1Msg, p2Msg, p2v \rangle
\end{array}
$$

$$
p(self) \triangleq entry(self) \lor P1S(self) \lor P1R(self) \lor P2S(self) \lor P2R(self)
$$

Allow infinite stuttering to prevent deadlock on termination.

$$
\begin{array}{lcl}
Terminating & \triangleq & \land\ \forall\, self \in ProcSet : pc[self] = \text{“Done”} \\
 & & \land\ \text{UNCHANGED } vars
\end{array}
$$

$$
\begin{array}{lcl}
Next & \triangleq & (\exists\, self \in Procs : p(self)) \\
 & & \quad \lor\ Terminating
\end{array}
$$

$$
\begin{array}{lcl}
Spec & \triangleq & \land\ Init \land \Box[Next]_{vars} \\
 & & \land\ \mathrm{WF}_{vars}(Next) \\
 & & \land\ \forall\, self \in Procs : \mathrm{WF}_{vars}(p(self))
\end{array}
$$

$$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{"Done"})$$

END TRANSLATION

$$Agreement \triangleq (\forall\, i,\, j \in Procs : decided[i] \neq -1 \land decided[j] \neq -1 \Rightarrow decided[i] = decided[j])$$
$$MinorityReport \triangleq (\exists\, j \in Procs : \text{TRUE} \Rightarrow (decided[j] = 1) \lor (decided[j] = -1))$$
$$Progress \triangleq (\exists\, j \in Procs : \text{TRUE} \Rightarrow \Diamond(decided[j] \neq -1))$$
$$BaitProgress \triangleq (\exists\, j \in Procs : \text{TRUE} \Rightarrow (decided[j] = -1))$$

Agreement:

We see that agreement is always satisfied, even when $N < 5$, $F < 5$, and $F > N/2$. According to our invariant property, we don't compare the initial $value(-1)$ as the process has not been decided.

We see that at any given time, no two processes decide differently. In all our testing we observe that agreement is held up.

Progress:

We see that progress property is always satisfied and the consensus is achieved. If we give the same preference values and vary the number of failures to zero or one for four nodes, the consensus is achieved and the program terminates which means that every process have decided a value not equal to -1. This is in *Sync* with the theoretical assumptions of the algorithm. When the case is $INPUT = \langle 0, 1, 1, 1 \rangle$ and there are no failures we see that the program terminates (Because the majority is present and processes k-lock -1, however, when we allow the number of failures as 1, the program doesn't terminate because the majority is not established.

Bait *Progress*:

We have defined a Bait *Progress* property which baits the model checker to find one process which has decided value as -1. The model checker using this invariant finds an execution where all the nodes have a decided $value \in \{0, 1\}$

Progress Violates, Bait *Progress* finds a way.

Consider example of $Input(INPUT)$ as $\langle 0, 0, 1, 1 \rangle$, $MaxRounds(r)$ as 3, $Nodes(N)$ as 4, $Failures(F)$ as 0. We see that here clearly there is no majority, the model checker will show a run where progress is not achieved and all the decided values are -1. By intuition, this is when the bit flips to the input itself, that is there are two zeros and two ones.

On the other hand, while checking Bait *Progress*, We see that the invariant breaks, that is The consensus is reached. The model checker presents a run where the bit flips in such a way that it gives a majority in the next round. Eventually, decided value is k-locked and the consensus is reached.

One particular Trace of consensus of zero is as *follows*(*Key Stages are given below*):

After certain executions, at $r = \langle 2, 2, 2, 1\rangle$ the third process flips its value to zero. The fourth process also flips the bit at $r = \langle 2, 2, 2, 2\rangle$. Thereby creating a majority Process one k-locks zero and moves to round 3, consequently, when all the processes reach round 3, the consensus is reached by
$decided = \langle 0, 0, 0, 0\rangle$

MINORITY Report:

Minority report is bait progress with consensus zero for a particular test case.

Minority report doesn't break when there are no $failures(F = 0)$ which is understandable as there is a majority. When $F = 1$ it is possible to get zero as consensus and the model

---

Minority Rounds.

When we have failures > 0, Interesting observation is that flipping of the bits takes at least 3 rounds to achieve zero as the consensus. For failure $= 1$, we observe this trace.

Intially, $Input(INPUT)$ as $\langle 0, 1, 1, 1\rangle$, $MaxRounds(r)$ as 3, $Nodes(N)$ as 4, $Failures(F)$ as 1. At $r = \langle 3, 2, 2, 1\rangle$ process 1 k-locks value as zero because of random bit flips to zero, which leads to the majority in the previous rounds. Therefore, all other processes after reaching round 3 have decided value as $k - 0$.