

```

┌────────────────────────────────── MODULE BenOr ───────────────────────────────────┐
EXTENDS Integers, TLC, Sequences, FiniteSets
CONSTANT N, F, MAXROUND, INPUT
  \ * N Nodes, F Failures
ASSUME  $N \in \text{Nat} \wedge F \in \text{Nat} \wedge F \leq N$ 
  \ * Assuming INPUT is a valid Sequence in the form  $\langle x, \dots, N \rangle$ 
Procs  $\triangleq 1 \dots N$ 
--fair algorithm BenOr
{
  variable p1Msg = {}, p2Msg = {};

  define
  {
    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)  $\triangleq \{m \in p1Msg : (m.round = r) \wedge (m.value = v)\}$ 
    ValueMsgP2(r, v)  $\triangleq \{m \in p2Msg : (m.round = r) \wedge (m.value = v)\}$ 
  }

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

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

    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  $\cup$   $\{[nodeid \mapsto self, round \mapsto r, value \mapsto i]\}$ 
}

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

    print("inside recieve");
    if ( Cardinality(ValueMsgP2(r, 1))  $\geq F + 1$  ) {
        decided := 1;
    }
    else if ( Cardinality(ValueMsgP2(r, 0))  $\geq F + 1$  ) {
        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))  $\geq 1$ ) ) {
        with ( v  $\in$  {0, 1} ) {
            p1v := v;
        } ;
    } ;
}

fair process ( p  $\in$  Procs )
variable r = 1, p1v = INPUT[self], p2v = -1, decided = -1;
{
    entry: while ( r < MAXROUND )
    {
        \ * SendP1  $\rightarrow$  macro which will post the value of that node to the message board as p1v
        P1S: SendP1(r, p1v);
        \ * RvcP1  $\rightarrow$  Get n-f values with p1v
        P1R: RvcP1(r);
        \ * SendP2  $\rightarrow$  Macro which sends the b value of the node to the message board as p2v
        P2S: SendP2(r, p2v);
        \ * RvcP2  $\rightarrow$  Get n-f values with p2v
        P2R: {

```

```

        RvcP2(r);
        r := r + 1;
      } ;
    } ;
  }
}

```

BEGIN TRANSLATION

VARIABLES $p1Msg, p2Msg, pc$

define statement

$ExtractValSet(S) \triangleq \{m.value : m \in S\}$
 $CollectP1Msgs(r) \triangleq \{m \in p1Msg : (m.round = r)\}$
 $CollectP2Msgs(r) \triangleq \{m \in p2Msg : (m.round = r)\}$
 $ValueMsg(r, v) \triangleq \{m \in p1Msg : (m.round = r) \wedge (m.value = v)\}$
 $ValueMsgP2(r, v) \triangleq \{m \in p2Msg : (m.round = r) \wedge (m.value = v)\}$

VARIABLES $r, p1v, p2v, decided$

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

$ProcSet \triangleq (Procs)$

$Init \triangleq$ **Global variables**

$\wedge p1Msg = \{\}$

$\wedge p2Msg = \{\}$

Process p

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

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

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

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

$\wedge pc = [self \in ProcSet \mapsto \text{"entry"}]$

$entry(self) \triangleq \wedge pc[self] = \text{"entry"}$
 $\wedge \text{IF } r[self] < MAXROUND$
 $\quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"P1S"}]$
 $\quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Done"}]$
 $\wedge \text{UNCHANGED } \langle p1Msg, p2Msg, r, p1v, p2v, decided \rangle$

$P1S(self) \triangleq \wedge pc[self] = \text{"P1S"}$
 $\wedge p1Msg' = (p1Msg \cup \{[nodeid \mapsto self, round \mapsto r[self], value \mapsto p1v[self]]\})$
 $\wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"P1R"}]$
 $\wedge \text{UNCHANGED } \langle p2Msg, r, p1v, p2v, decided \rangle$

$P1R(self) \triangleq \wedge pc[self] = \text{"P1R"}$
 $\wedge (Cardinality(CollectP1Msgs(r[self])) \geq N - F)$
 $\wedge \text{IF } (Cardinality(ValueMsg(r[self], 1)) * 2 > N)$

$$\begin{aligned}
& \text{THEN } \wedge p2v' = [p2v \text{ EXCEPT } ![self] = 1] \\
& \text{ELSE } \wedge \text{IF } (Cardinality(ValueMsg(r[self], 0)) * 2 > N) \\
& \quad \text{THEN } \wedge p2v' = [p2v \text{ EXCEPT } ![self] = 0] \\
& \quad \text{ELSE } \wedge p2v' = [p2v \text{ EXCEPT } ![self] = -1] \\
& \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"P2S"}] \\
& \wedge \text{UNCHANGED } \langle p1Msg, p2Msg, r, p1v, decided \rangle \\
P2S(self) & \triangleq \wedge pc[self] = \text{"P2S"} \\
& \wedge p2Msg' = (p2Msg \cup \{[nodeid \mapsto self, round \mapsto r[self], value \mapsto p2v[self]]\}) \\
& \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"P2R"}] \\
& \wedge \text{UNCHANGED } \langle p1Msg, r, p1v, p2v, decided \rangle \\
P2R(self) & \triangleq \wedge pc[self] = \text{"P2R"} \\
& \wedge (Cardinality(CollectP2Msgs(r[self])) \geq N - F) \\
& \wedge \text{IF } Cardinality(ValueMsgP2(r[self], 1)) \geq F + 1 \\
& \quad \text{THEN } \wedge decided' = [decided \text{ EXCEPT } ![self] = 1] \\
& \quad \text{ELSE } \wedge \text{IF } Cardinality(ValueMsgP2(r[self], 0)) \geq F + 1 \\
& \quad \quad \text{THEN } \wedge decided' = [decided \text{ EXCEPT } ![self] = 0] \\
& \quad \quad \text{ELSE } \wedge \text{TRUE} \\
& \quad \quad \wedge \text{UNCHANGED } decided \\
& \wedge \text{IF } (Cardinality(ValueMsgP2(r[self], 0)) > 0) \\
& \quad \text{THEN } \wedge p1v' = [p1v \text{ EXCEPT } ![self] = 0] \\
& \quad \text{ELSE } \wedge \text{IF } (Cardinality(ValueMsgP2(r[self], 1)) > 0) \\
& \quad \quad \text{THEN } \wedge p1v' = [p1v \text{ EXCEPT } ![self] = 1] \\
& \quad \quad \text{ELSE } \wedge \text{IF } (Cardinality(ValueMsgP2(r[self], -1)) \geq 1) \\
& \quad \quad \quad \text{THEN } \wedge \exists v \in \{0, 1\} : \\
& \quad \quad \quad \quad p1v' = [p1v \text{ EXCEPT } ![self] = v] \\
& \quad \quad \quad \text{ELSE } \wedge \text{TRUE} \\
& \quad \quad \quad \wedge p1v' = p1v \\
& \wedge r' = [r \text{ EXCEPT } ![self] = r[self] + 1] \\
& \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"entry"}] \\
& \wedge \text{UNCHANGED } \langle p1Msg, p2Msg, p2v \rangle \\
p(self) & \triangleq entry(self) \vee P1S(self) \vee P1R(self) \vee P2S(self) \vee P2R(self) \\
& \text{Allow infinite stuttering to prevent deadlock on termination.} \\
Terminating & \triangleq \wedge \forall self \in ProcSet : pc[self] = \text{"Done"} \\
& \wedge \text{UNCHANGED } vars \\
Next & \triangleq (\exists self \in Procs : p(self)) \\
& \vee Terminating \\
Spec & \triangleq \wedge Init \wedge \Box [Next]_{vars} \\
& \wedge WF_{vars}(Next) \\
& \wedge \forall self \in Procs : WF_{vars}(p(self)) \\
Termination & \triangleq \Diamond (\forall self \in ProcSet : pc[self] = \text{"Done"})
\end{aligned}$$

END TRANSLATION

$Agreement \triangleq (\forall i, j \in Procs : decided[i] \neq -1 \wedge decided[j] \neq -1 \Rightarrow decided[i] = decided[j])$
 $MinorityReport \triangleq (\exists j \in Procs : TRUE \Rightarrow (decided[j] = 1) \vee (decided[j] = -1))$
 $Progress \triangleq (\exists j \in Procs : TRUE \Rightarrow \Diamond(decided[j] \neq -1))$
 $BaitProgress \triangleq (\exists j \in Procs : TRUE \Rightarrow (decided[j] = -1))$

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.

Progress:

We see that progress property is always satisfied and consensus is achieved. If we give same preference values and vary the number of failures to zero or one for four nodes, 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 algorithm. When the case is $INPUT = \langle 0, 1, 1, 1 \rangle$ and there are no failures we see that the program terminates (Because 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 have 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 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 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, 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 atleast 3 rounds to achieve zero as the consensus. For failure $= 1$, we observe this trace.

Initially, *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 majority in the previous rounds. Therefore, all other processes after reaching round 3 have decided value as $k - 0$.