MODULE *Pactus*

The specification of the Pactus consensus algorithm :
https://pactus.org/learn/consensus/protocol/

EXTENDS *Integers*, *Sequences*, *FiniteSets*, *TLC*

CONSTANT

The maximum number of height.
this is to restrict the allowed behaviours that TLC scans through.

MaxHeight,

The maximum number of round per height.
this is to restrict the allowed behaviours that TLC scans through.

MaxRound,

The maximum number of cp − round per height.
this is to restrict the allowed behaviours that TLC scans through.

MaxCPRound,

The total number of faulty nodes

NumFaulty,

The index of faulty nodes

FaultyNodes

VARIABLES

log is a set of received messages in the system.

log,

states represents the state of each replica in the consensus protocol.

states

Total number of replicas, which is $3f+1$, where f is the number of faulty nodes.

$Replicas \triangleq (3 * NumFaulty) + 1$

Quorum is $2/3+$ of total replicas that is $2f+1$

$Quorum \triangleq (2 * NumFaulty) + 1$

OneThird is $1/3+$ of total replicas that is $f+1$

$OneThird \triangleq NumFaulty + 1$

A tuple with all variables in the spec (for ease of use in temporal conditions)

$vars \triangleq \langle states, log \rangle$

ASSUME

$\land\ NumFaulty \geq 1$

$\land\ FaultyNodes \subseteq 0 \, .. \, Replicas - 1$

Helper functions

Fetch a subset of messages in the network based on the *params* filter.

$SubsetOfMsgs(params) \triangleq$

$\{msg \in log : \forall field \in \text{DOMAIN } params : msg[field] = params[field]\}$

IsProposer checks if the replica is the proposer for this round.
To simplify, we assume the proposer always starts with the first replica,
and moves to the next by the change-proposer phase.

```
$IsProposer(index) \triangleq$
    $states[index].round \% Replicas = index$
```

Helper function to check if a node is faulty or not.

```
$IsFaulty(index) \triangleq index \in FaultyNodes$
```

HasPrepareQuorum checks if there is a quorum of
the *PREPARE* votes in this round.

```
$HasPrepareQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "PREPARE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto 0])) \geq Quorum$
```

HasPrecommitQuorum checks if there is a quorum of
the *PRECOMMIT* votes in this round.

```
$HasPrecommitQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "PRECOMMIT",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto 0])) \geq Quorum$

$CPHasPreVotesQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "CP:PRE-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round])) \geq Quorum$

$CPHasPreVotesQuorumForOne(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "CP:PRE-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \mapsto 1])) \geq Quorum$

$CPHasPreVotesQuorumForZero(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "CP:PRE-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \quad\; \mapsto 0])) \geq Quorum$

$CPHasPreVotesForZeroAndOne(index) \triangleq$
    $\land\ Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:PRE-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \quad\; \mapsto 0])) \geq 1$
    $\land\ Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:PRE-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \quad\; \mapsto 1])) \geq 1$

$CPHasOneMainVotesZeroInPrvRound(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:MAIN-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round - 1,$
        $cp\_val \quad\; \mapsto 0])) > 0$

$CPHasOneMainVotesOneInPrvRound(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:MAIN-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round - 1,$
        $cp\_val \quad\; \mapsto 1])) > 0$

$CPAllMainVotesAbstainInPrvRound(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:MAIN-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round - 1,$
        $cp\_val \quad\; \mapsto 2])) \geq Quorum$


$CPHasMainVotesQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \qquad\;\; \mapsto$ “CP:MAIN-VOTE”,
        $height \qquad \mapsto states[index].height,$
        $round \qquad \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round])) \geq Quorum$

$CPHasMainVotesQuorumForOne(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "CP:MAIN-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \mapsto 1])) \geq Quorum$

$CPHasMainVotesQuorumForZero(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "CP:MAIN-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \mapsto 0])) \geq Quorum$

$GetProposal(height, round) \triangleq$
    $SubsetOfMsgs([type \mapsto$ "PROPOSAL", $height \mapsto height, round \mapsto round])$

$HasProposal(index) \triangleq$
    $Cardinality(GetProposal(states[index].height, states[index].round)) > 0$

$HasBlockAnnounce(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ "BLOCK-ANNOUNCE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $cp\_round \mapsto 0,$
        $cp\_val \mapsto 0])) \geq 1$

  Helper function to check if the block is committed or not.
  A block is considered committed iff supermajority of non-faulty replicas announce the same block.
$IsCommitted(height) \triangleq$
    LET $subset \triangleq SubsetOfMsgs([$
          $type \mapsto$ "BLOCK-ANNOUNCE",
          $height \mapsto height,$
          $cp\_round \mapsto 0,$
          $cp\_val \mapsto 0])$
    IN  $\land Cardinality(subset) \geq Quorum$
        $\land \forall m1, m2 \in subset : m1.round = m2.round$
```

Network functions

SendMsg simulates a replica sending a message by appending it to the log

$SendMsg(msg) \triangleq$
$\quad log' = log \cup msg$

SendProposal is used to broadcast the PROPOSAL into the network.

```
$SendProposal(index) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "PROPOSAL",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto index,$
        $cp\_round \mapsto 0,$
        $cp\_val \mapsto 0]\})$
```

SendPrepareVote is used to broadcast PREPARE votes into the network.

```
$SendPrepareVote(index) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "PREPARE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto index,$
        $cp\_round \mapsto 0,$
        $cp\_val \mapsto 0]\})$
```

SendPrecommitVote is used to broadcast PRECOMMIT votes into the network.

```
$SendPrecommitVote(index) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "PRECOMMIT",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto index,$
        $cp\_round \mapsto 0,$
        $cp\_val \mapsto 0]\})$
```

SendCPPreVote is used to broadcast $CP : PRE - VOTE$ votes into the network.

```
$SendCPPreVote(index, cp\_val) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "CP:PRE-VOTE",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto index,$
        $cp\_round \mapsto states[index].cp\_round,$
        $cp\_val \mapsto cp\_val]\})$
```

SendCPMainVote is used to broadcast $CP : MAIN - VOTE$ votes into the network.

```
$SendCPMainVote(index, cp\_val) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "CP:MAIN-VOTE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$,
        $cp\_round \mapsto states[index].cp\_round$,
        $cp\_val \mapsto cp\_val]\})$

$SendCPVotesForNextRound(index, cp\_val) \triangleq$
    $SendMsg(\{$
    $[$
        $type \mapsto$ "CP:PRE-VOTE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$,
        $cp\_round \mapsto states[index].cp\_round + 1$,
        $cp\_val \mapsto cp\_val]$,
    $[$
        $type \mapsto$ "CP:MAIN-VOTE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$,
        $cp\_round \mapsto states[index].cp\_round + 1$,
        $cp\_val \mapsto cp\_val]\})$

  AnnounceBlock is used to broadcast BLOCK − ANNOUNCE messages into the network.
$AnnounceBlock(index) \triangleq$
    $SendMsg(\{[$
        $type \mapsto$ "BLOCK-ANNOUNCE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$,
        $cp\_round \mapsto 0$,
        $cp\_val \mapsto 0]\})$
```

States functions

```
  NewHeight state
$NewHeight(index) \triangleq$
    IF $states[index].height \geq MaxHeight$
     THEN UNCHANGED $\langle states, log \rangle$
     ELSE
        $\land \neg IsFaulty(index)$
        $\land states[index].name =$ "new-height"
        $\land states[index].height < MaxHeight$
        $\land states' = [states$ EXCEPT
        $![index].name = $ "propose",
        $![index].height = states[index].height + 1$,
        $![index].round = 0]$
      $\land$ UNCHANGED $\langle log \rangle$
```

Propose state

```
Propose(index) $\triangleq$
    $\land$ $\neg IsFaulty(index)$
    $\land$ $states[index].name = $ "propose"
    $\land$ IF $IsProposer(index)$
          THEN $SendProposal(index)$
          ELSE UNCHANGED $\langle log \rangle$
    $\land$ $states' = [states$ EXCEPT
          $![index].name = $ "prepare",
          $![index].timeout = $ FALSE,
          $![index].cp\_round = 0]$
```

Prepare state

```
Prepare(index) $\triangleq$
    $\land$ $\neg IsFaulty(index)$
    $\land$ $states[index].name = $ "prepare"
    $\land$ IF $HasPrepareQuorum(index)$
          THEN $\land$ $states' = [states$ EXCEPT $![index].name = $ "precommit"$]$
               $\land$ UNCHANGED $\langle log \rangle$
          ELSE $\land$ $HasProposal(index)$
               $\land$ $SendPrepareVote(index)$
               $\land$ UNCHANGED $\langle states \rangle$
```

Precommit state

```
Precommit(index) $\triangleq$
    $\land$ $\neg IsFaulty(index)$
    $\land$ $states[index].name = $ "precommit"
    $\land$ IF $HasPrecommitQuorum(index)$
        THEN $\land$ $states' = [states$ EXCEPT $![index].name = $ "commit"$]$
             $\land$ UNCHANGED $\langle log \rangle$
        ELSE $\land$ $HasProposal(index)$
             $\land$ $SendPrecommitVote(index)$
             $\land$ UNCHANGED $\langle states \rangle$
```

Commit state

```
Commit(index) $\triangleq$
    $\land$ $\neg IsFaulty(index)$
    $\land$ $states[index].name = $ "commit"
    $\land$ $AnnounceBlock(index)$
    $\land$ $states' = [states$ EXCEPT
```

```
        $![index].name = $ “new-height” $]$

$Timeout : A\ non-faulty\ Replica\ try\ to\ change\ the\ proposer\ if\ its\ timer\ expires.$
$Timeout(index) \triangleq$
    $\land$  $\neg IsFaulty(index)$
    $\land$  $states[index].round < MaxRound$
    $\land$  $states[index].timeout = \text{FALSE}$
    $\land$
        $\lor$
            $\land$ $states[index].name = $ “prepare”
            $\land$ $SendCPPreVote(index, 1)$

        $\lor$
            $\land$ $states[index].name = $ “precommit”
            $\land$ $SendCPPreVote(index, 0)$
    $\land$ $states' = [states$ EXCEPT
            $![index].name = $ “cp:main-vote”,
            $![index].timeout = \text{TRUE}]$


$CPPreVote(index) \triangleq$
    $\land \neg IsFaulty(index)$
    $\land$ $states[index].name = $ “cp:pre-vote”
    $\land$
        $\lor$
            $\land$ $CPHasOneMainVotesOneInPrvRound(index)$
            $\land$ $SendCPPreVote(index, 1)$
        $\lor$
            $\land$ $CPHasOneMainVotesZeroInPrvRound(index)$
            $\land$ $SendCPPreVote(index, 0)$
        $\lor$
            $\land$ $CPAllMainVotesAbstainInPrvRound(index)$
            $\land$ $SendCPPreVote(index, 0)$  $biased\ to\ zero$
    $\land$ $states' = [states$ EXCEPT $![index].name = $ “cp:main-vote” $]$

$CPMainVote(index) \triangleq$
    $\land \neg IsFaulty(index)$
    $\land$ $states[index].name = $ “cp:main-vote”
    $\land$ $CPHasPreVotesQuorum(index)$
    $\land$
        $\lor$
              $all\ votes\ for\ 1$
            $\land$ $CPHasPreVotesQuorumForOne(index)$
            $\land$ $SendCPMainVote(index, 1)$
            $\land$ $states' = [states$ EXCEPT $![index].name = $ “cp:decide” $]$
    $\lor$
            all votes for 0
        $\land$ $CPHasPreVotesQuorumForZero(index)$
        $\land$ $SendCPMainVote(index, 0)$
        $\land$ $states' = [states$ EXCEPT $![index].name =$ "cp:decide"$]$
    $\lor$
            Abstain vote
        $\land$ $CPHasPreVotesForZeroAndOne(index)$
        $\land$ $SendCPMainVote(index, 2)$
        $\land$ $states' = [states$ EXCEPT $![index].name =$ "cp:decide"$]$

$CPDecide(index) \triangleq$
  $\land$ $\neg IsFaulty(index)$
  $\land$ $states[index].name =$ "cp:decide"
  $\land$
    $\lor$
        $\land$ $states[index].cp\_decided = 1$
        $\land$ $states' = [states$ EXCEPT $![index].name =$ "propose"$]$
    $\lor$
        $\land$ $states[index].cp\_decided = 0$
        $\land$ $states' = [states$ EXCEPT $![index].name =$ "prepare"$]$
    $\lor$
        $\land$ $states[index].cp\_decided = -1$
        $\land$ $CPHasMainVotesQuorum(index)$
        $\land$
          IF   $CPHasMainVotesQuorumForOne(index)$
           THEN $states' = [states$ EXCEPT $![index].name =$ "cp:pre-vote",
                                          $![index].cp\_decided = 1,$
                                          $![index].cp\_round = states[index].cp\_round + 1]$
           ELSE IF $\lor$ $CPHasMainVotesQuorumForZero(index)$
                   $\lor$ $states[index].cp\_round = MaxCPRound$
               THEN $states' = [states$ EXCEPT $![index].name =$ "cp:pre-vote",
                                              $![index].cp\_decided = 0,$
                                              $![index].cp\_round = states[index].cp\_round + 1]$
               ELSE $states' = [states$ EXCEPT $![index].name =$ "cp:pre-vote",
                                              $![index].cp\_round = states[index].cp\_round + 1]$

  $\land$ $log' = log$

$Sync(index) \triangleq$
   $\land$ $\neg IsFaulty(index)$
   $\land$
      $\lor$ $states[index].name =$ "cp:pre-vote"
      $\lor$ $states[index].name =$ "cp:main-vote"
      $\lor$ $states[index].name =$ "cp:decide"
```

```
        $\land\ HasBlockAnnounce(index)$
        $\land\ states' = [states\ \text{EXCEPT}\ ![index].name = \text{“prepare”}]$
        $\land\ log' = log$
```

```
$Init\ \triangleq$
    $\land\ log = \{\}$
    $\land\ states = [index \in 0\ ..\ Replicas - 1 \mapsto [$
        $name\qquad\ \mapsto \text{“new-height”},$
        $height\qquad \mapsto 0,$
        $round\qquad\ \mapsto 0,$
        $timeout\quad\ \mapsto \text{FALSE},$
        $cp\_round\ \ \mapsto 0,$
        $cp\_decided \mapsto -1]]$

$Next\ \triangleq$
    $\exists\, index \in 0\ ..\ Replicas - 1 :$
        $\lor\ NewHeight(index)$
        $\lor\ Propose(index)$
        $\lor\ Prepare(index)$
        $\lor\ Precommit(index)$
        $\lor\ Timeout(index)$
        $\lor\ Commit(index)$
        $\lor\ Sync(index)$
        $\lor\ CPPreVote(index)$
        $\lor\ CPMainVote(index)$
        $\lor\ CPDecide(index)$

$Spec\ \triangleq$
    $Init \land \Box[Next]_{vars} \land \text{WF}_{vars}(Next)$
```

Success : All non − faulty nodes eventually commit at MaxHeight.

```
$Success\ \triangleq\ \Diamond(IsCommitted(MaxHeight))$
```

TypeOK is the type − correctness invariant.

```
$TypeOK\ \triangleq$
    $\land\quad \forall\, index \in 0\ ..\ Replicas - 1 :$
            $\land\ states[index].name \in \{\text{“new-height”}, \text{“propose”}, \text{“prepare”},$
                $\text{“precommit”}, \text{“commit”}, \text{“cp:pre-vote”}, \text{“cp:main-vote”}, \text{“cp:decide”}\}$
            $\land\ states[index].height \leq MaxHeight$
            $\land\ states[index].round \leq MaxRound$
            $\land\ states[index].cp\_round \leq MaxCPRound + 1$
            $\land\ states[index].name = \text{“new-height”} \land states[index].height > 1 \Rightarrow$
                $\land\ IsCommitted(states[index].height - 1)$
```

```
∧ $states[index].name = $ "precommit" $\Rightarrow$
    ∧ $HasPrepareQuorum(index)$
    ∧ $HasProposal(index)$
∧ $states[index].name = $ "commit" $\Rightarrow$
    ∧ $HasPrepareQuorum(index)$
    ∧ $HasPrecommitQuorum(index)$
    ∧ $HasProposal(index)$
∧ $\forall\, round \in 0 \,..\, states[index].round :$
    Not more than one proposal per round
    ∧ $Cardinality(GetProposal(states[index].height,\ round)) \leq 1$
```