MODULE *Raft*

This is the formal specification for the *Raft* consensus algorithm.

EXTENDS *Naturals*, *FiniteSets*, *Sequences*, *TLC*, *Client*, *Server*, *Messages*

VARIABLE *allStates*

All variables; used for stuttering (asserting state hasn't changed).

$vars \triangleq \langle allStates, messages, clientVars, serverVars, stateVars, followerVars, candidateVars, leaderVars, log$

$Init \triangleq$
- $\land\ InitClientVars$
- $\land\ InitServerVars$
- $\land\ InitMessageVars$
- $\land\ allStates = 0$

Defines how the variables may transition.

$Next \triangleq$
- $\land \lor \exists\, i \in Server : Restart(i)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i \in Server : TimeoutFollower(i)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i \in Server : TimeoutCandidate(i)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i, j \in Server : RequestPreVote(i, j)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i, j \in Server : RequestVote(i, j)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i \in Server : BecomeCandidate(i)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i \in Server : BecomeLeader(i)$
  - $\land$ UNCHANGED $\langle clientVars \rangle$
- $\lor \exists\, i \in Client, j \in Server : OpenSession(i, j)$
  - $\land$ UNCHANGED $\langle serverVars, stateVars, followerVars, candidateVars, leaderVars, logVars \rangle$
- $\lor \exists\, i \in Client, j \in Server : CloseSession(i, j)$
  - $\land$ UNCHANGED $\langle serverVars, stateVars, followerVars, candidateVars, leaderVars, logVars \rangle$
- $\lor \exists\, i \in Client, j \in Server : ClientRequest(i, j)$
  - $\land$ UNCHANGED $\langle serverVars, stateVars, followerVars, candidateVars, leaderVars, logVars \rangle$
- $\lor \exists\, i \in Server : AdvanceCommitIndex(i)$

$\land$ UNCHANGED $\langle clientVars \rangle$
$\lor \exists\, i \in Server : ApplyEntry(i)$
$\land$ UNCHANGED $\langle clientVars \rangle$
$\lor \exists\, i, j \in Server : AppendEntries(i, j)$
$\land$ UNCHANGED $\langle clientVars \rangle$
$\lor \exists\, m \in$ DOMAIN $messages : Receive(m)$
$\land$ UNCHANGED $\langle clientVars \rangle$
$\lor \exists\, m \in$ DOMAIN $messages : DuplicateMessage(m)$
$\land$ UNCHANGED $\langle serverVars, stateVars, followerVars, candidateVars, leaderVars, logVars, clientVars$
$\lor \exists\, m \in$ DOMAIN $messages : DropMessage(m)$
$\land$ UNCHANGED $\langle serverVars, stateVars, followerVars, candidateVars, leaderVars, logVars, clientVars$
$\land\, allStates' = allStates + 1$

$Inv \triangleq$
$\land \exists\, s1, s2 \in Server :$
$Len(log[s1]) > 0 \land Len(log[s2]) > 0 \Rightarrow \exists\, l1 \in$ DOMAIN $log[s1], l2 \in$ DOMAIN $log[s2] :$
$l1 \leq commitIndex[s1] \land l2 \leq commitIndex[s2] \Rightarrow log[s1][l1] = log[s2][l2]$

The specification must start with the initial state and transition according to *Next*.

$Spec \triangleq Init \land \Box[Next]_{vars}$