──────────────────── MODULE *paxoscommit* ────────────────────

This specification is discussed in *"Paxos Commit"*, Lecture 6 of the TLA+ Video Course.

This module specifies the *Paxos Commit* algorithm. We specify only safety properties, not liveness properties. We simplify the specification in the following ways.

 - As in the specification of module *TwoPhase*, and for the same reasons, we let the variable *msgs* be the set of all messages that have ever been sent. If a message is sent to a set of recipients, only one copy of the message appears in *msgs*.

 - We do not explicitly model the receipt of messages. If an operation can be performed when a process has received a certain set of messages, then the operation is represented by an action that is enabled when those messages are in the set *msgs* of sent messages. (We are specifying only safety properties, which assert what events can occur, and the operation can occur if the messages that enable it have been sent.)

 - We do not model leader selection. We define actions that the current leader may perform, but do not specify who performs them.

As in the specification of Two-Phase commit in module *TwoPhase*, we have *RMs* spontaneously issue Prepared messages and we ignore Prepare messages.

EXTENDS *Integers*

$Maximum(S) \triangleq$

If S is a set of numbers, then this define $Maximum(S)$ to be the maximum of those numbers, or -1 if S is empty.

IF $S = \{\}$ THEN -1
       ELSE CHOOSE $n \in S : \forall\, m \in S : n \geq m$

CONSTANT RM,   The set of resource managers.
          $Acceptor$,   The set of acceptors.
          $Majority$,   The set of majorities of acceptors
          $Ballot$ The set of ballot numbers

We assume the following properties of the declared constants.

ASSUME
   $\wedge\ Ballot \subseteq Nat$
   $\wedge\ 0 \in Ballot$
   $\wedge\ Majority \subseteq$ SUBSET $Acceptor$
   $\wedge\ \forall\, MS1,\, MS2 \in Majority : MS1 \cap MS2 \neq \{\}$

      All we assume about the set *Majority* of majorities is that any two majorities have non-empty intersection.

$Messages \triangleq$

The set of all possible messages. There are messages of type **"Commit"** and **"Abort"** to announce the decision, as well as messages for each phase of each instance of *ins* of the *Paxos* consensus algorithm. The *acc* field indicates the sender of a message from an acceptor to the leader; messages from a leader are broadcast to all acceptors.

$[type : \{\,\text{"phase1a"}\,\},\ ins : RM,\ bal : Ballot \setminus \{0\}]$
    $\cup$
$[type : \{\,\text{"phase1b"}\,\},\ ins : RM,\ mbal : Ballot,\ bal : Ballot \cup \{-1\},$

1

$$val : \{\text{``prepared''}, \text{``aborted''}, \text{``none''}\}, acc : Acceptor]$$
$$\cup$$
$$[type : \{\text{``phase2a''}\}, ins : RM, bal : Ballot, val : \{\text{``prepared''}, \text{``aborted''}\}]$$
$$\cup$$
$$[type : \{\text{``phase2b''}\}, acc : Acceptor, ins : RM, bal : Ballot,$$
$$val : \{\text{``prepared''}, \text{``aborted''}\}]$$
$$\cup$$
$$[type : \{\text{``Commit''}, \text{``Abort''}\}]$$

VARIABLES

$rmState$,	$rmState[r]$ is the state of resource manager r.
$aState$,	$aState[ins][ac]$ is the state of acceptor ac for instance ins of the *Paxos* algorithm.
$msgs$	The set of all messages ever sent.

$PCTypeOK \triangleq$

The type-correctness invariant. Each acceptor maintains the values *mbal*, *bal*, and *val* for each instance of the *Paxos* consensus algorithm.

$\wedge\ rmState \in [RM \rightarrow \{\text{``working''}, \text{``prepared''}, \text{``committed''}, \text{``aborted''}\}]$
$\wedge\ aState\ \ \in [RM \rightarrow [Acceptor \rightarrow [mbal : Ballot,$
$bal\ \ \ : Ballot \cup \{-1\},$
$val\ \ \ : \{\text{``prepared''}, \text{``aborted''}, \text{``none''}\}]]]$
$\wedge\ msgs \subseteq Messages$

$PCInit \triangleq$ The initial predicate.
$\wedge\ rmState = [r \in RM \mapsto \text{``working''}]$
$\wedge\ aState\ \ = [r \in RM \mapsto$
$[ac \in Acceptor$
$\mapsto [mbal \mapsto 0,\ bal\ \mapsto -1,\ val\ \mapsto \text{``none''}]]]$
$\wedge\ msgs = \{\}$

THE ACTIONS

$Send(m) \triangleq msgs' = msgs \cup \{m\}$

An action expression that describes the sending of message m.

RM ACTIONS

$RMPrepare(r) \triangleq$

Resource manager r prepares by sending a phase $2a$ message for ballot number 0 with value "prepared".

$\wedge\ rmState[r] = \text{``working''}$
$\wedge\ rmState' = [rmState \text{ EXCEPT } ![r] = \text{``prepared''}]$
$\wedge\ Send([type \mapsto \text{``phase2a''}, ins \mapsto r, bal \mapsto 0, val \mapsto \text{``prepared''}])$
$\wedge\ \text{UNCHANGED } aState$

$RMChooseToAbort(r) \triangleq$

2

$\wedge\ rmState[r] =$ "working"
$\wedge\ rmState' = [rmState$ EXCEPT $![r] =$ "aborted"$]$
$\wedge\ Send([type \mapsto$ "phase2a", $ins \mapsto r,\ bal \mapsto 0,\ val \mapsto$ "aborted"$])$
$\wedge$ UNCHANGED $aState$

$RMRcvCommitMsg(r)\ \triangleq$

$\wedge\ [type \mapsto$ "Commit"$] \in msgs$
$\wedge\ rmState' = [rmState$ EXCEPT $![r] =$ "committed"$]$
$\wedge$ UNCHANGED $\langle aState,\ msgs \rangle$

$RMRcvAbortMsg(r)\ \triangleq$

$\wedge\ [type \mapsto$ "Abort"$] \in msgs$
$\wedge\ rmState' = [rmState$ EXCEPT $![r] =$ "aborted"$]$
$\wedge$ UNCHANGED $\langle aState,\ msgs \rangle$

LEADER ACTIONS

$Phase1a(bal,\ r)\ \triangleq$

$\wedge\ Send([type \mapsto$ "phase1a", $ins \mapsto r,\ bal \mapsto bal])$
$\wedge$ UNCHANGED $\langle rmState,\ aState \rangle$

$Phase2a(bal,\ r)\ \triangleq$

$\wedge\ \neg\exists\, m \in msgs : \wedge\ m.type =$ "phase2a"
$\qquad\qquad\qquad\qquad\ \wedge\ m.bal = bal$
$\qquad\qquad\qquad\qquad\ \wedge\ m.ins = r$
$\wedge\ \exists\, MS \in Majority :$

$$
\begin{aligned}
\text{LET } mset \triangleq \{m \in msgs : &\land m.type = \text{``phase1b''}\\
&\land m.ins = r\\
&\land m.mbal = bal\\
&\land m.acc \in MS\}
\end{aligned}
$$

$$
\begin{aligned}
maxbal &\triangleq Maximum(\{m.bal : m \in mset\})\\
val &\triangleq \text{IF } maxbal = -1\\
&\qquad \text{THEN ``aborted''}\\
&\qquad \text{ELSE } (\text{CHOOSE } m \in mset : m.bal = maxbal).val
\end{aligned}
$$

$$
\begin{aligned}
\text{IN} \quad &\land \forall ac \in MS : \exists m \in mset : m.acc = ac\\
&\land Send([type \mapsto \text{``phase2a''}, ins \mapsto r, bal \mapsto bal, val \mapsto val])\\
&\land \text{UNCHANGED } \langle rmState, aState \rangle
\end{aligned}
$$

$PCDecide \triangleq$

A leader can decide that *Paxos Commit* has reached a result and send a message announcing the result if it has received the necessary phase $2b$ messages.

$\land \text{LET } Decided(r, v) \triangleq$

True iff instance r of the *Paxos* consensus algorithm has chosen the value v.

$$
\begin{aligned}
&\exists b \in Ballot, MS \in Majority :\\
&\quad \forall ac \in MS : [type \mapsto \text{``phase2b''}, ins \mapsto r,\\
&\qquad\qquad\qquad\quad bal \mapsto b, val \mapsto v, acc \mapsto ac] \in msgs
\end{aligned}
$$

$$
\begin{aligned}
\text{IN} \quad \lor \; &\land \forall r \in RM : Decided(r, \text{``prepared''})\\
&\land Send([type \mapsto \text{``Commit''}])\\
\lor \; &\land \exists r \in RM : Decided(r, \text{``aborted''})\\
&\land Send([type \mapsto \text{``Abort''}])
\end{aligned}
$$

$\land \text{UNCHANGED } \langle rmState, aState \rangle$

ACCEPTOR ACTIONS

$Phase1b(acc) \triangleq$

$\exists m \in msgs :$

$\quad \land m.type = \text{``phase1a''}$

$\quad \land aState[m.ins][acc].mbal < m.bal$

$\quad \land aState' = [aState \text{ EXCEPT } ![m.ins][acc].mbal = m.bal]$

$$
\begin{aligned}
\quad \land Send([&type \mapsto \text{``phase1b''},\\
&ins \mapsto m.ins,\\
&mbal \mapsto m.bal,\\
&bal \mapsto aState[m.ins][acc].bal,\\
&val \mapsto aState[m.ins][acc].val,\\
&acc \mapsto acc])
\end{aligned}
$$

$\quad \land \text{UNCHANGED } rmState$

$Phase2b(acc) \triangleq$

$\land \exists m \in msgs :$

$\quad \land m.type = \text{``phase2a''}$

$\quad \land aState[m.ins][acc].mbal \leq m.bal$

$\quad \land aState' = [aState \text{ EXCEPT } ![m.ins][acc].mbal = m.bal,$

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad\quad\quad\quad ![m.ins][acc].bal \quad = m.bal, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad ![m.ins][acc].val \quad = m.val] \\
\quad\wedge\, Send([type \mapsto \text{``phase2b''}, ins \mapsto m.ins,\, bal \mapsto m.bal, \\
\quad\quad\quad\quad\quad val \mapsto m.val,\, acc \mapsto acc]) \\
\wedge\, \textsc{unchanged}\ rmState
\end{array}
$$

$PCNext \triangleq$ The next-state action
$$
\begin{array}{l}
\vee\, \exists\, r \in RM :\ \vee\, RMPrepare(r) \\
\quad\quad\quad\quad\quad\quad \vee\, RMChooseToAbort(r) \\
\quad\quad\quad\quad\quad\quad \vee\, RMRcvCommitMsg(r) \\
\quad\quad\quad\quad\quad\quad \vee\, RMRcvAbortMsg(r) \\
\vee\, \exists\, bal \in Ballot \setminus \{0\},\, r \in RM : Phase1a(bal,\, r) \vee Phase2a(bal,\, r) \\
\vee\, PCDecide \\
\vee\, \exists\, acc \in Acceptor : Phase1b(acc) \vee Phase2b(acc)
\end{array}
$$

The following part of the spec is not covered in Lecture 7. It will be explained in Lecture 8.

$PCSpec \triangleq PCInit \wedge \Box[PCNext]_{\langle rmState,\, aState,\, msgs \rangle}$

The complete spec of the *Paxos Commit* protocol.

$\textsc{theorem}\ PCSpec \Rightarrow \Box PCTypeOK$

We now assert that the *Paxos Commit* protocol implements the transaction commit protocol of module *TCommit*. The following statement imports into the current module the definitions from module *TCommit*, including the definition of *TCSpec*.

$\textsc{instance}\ TCommitOriginal$

$\textsc{theorem}\ PCSpec \Rightarrow TCSpec$