MODULE *Paxos*

This is a specification of the *Paxos* protocol without explicit leaders or learners.

EXTENDS TLC, $Naturals$, $FiniteSets$, $Integers$

CONSTANTS any, $none$, $Replicas$, $Values$, $Ballots$, $Quorums$

VARIABLES $messages$ Set of all messages sent.
VARIABLES $decision$ Decided value of an acceptor.
VARIABLES $maxBallot$ Maximum ballot an acceptor has seen.
VARIABLES $maxVBallot$ Maximum ballot an acceptor has accepted.
VARIABLES $maxValue$ Maximum value an acceptor has accepted.

Set of all possible messages.

$$\begin{aligned}
P1aMessage &\triangleq [type : \{\text{"P1a"}\},\\
&\qquad ballot : Ballots \setminus \{0\}]\\
P1bMessage &\triangleq [type : \{\text{"P1b"}\},\\
&\qquad ballot : Ballots,\\
&\qquad acceptor : Replicas,\\
&\qquad maxVBallot : Ballots,\\
&\qquad maxValue : Values \cup \{none\}] \quad (maxVBallot = 0) \equiv (maxValue = none)\\
P2aMessage &\triangleq [type : \{\text{"P2a"}\},\\
&\qquad ballot : Ballots,\\
&\qquad value : Values \cup \{any\}]\\
P2bMessage &\triangleq [type : \{\text{"P2b"}\},\\
&\qquad ballot : Ballots,\\
&\qquad acceptor : Replicas,\\
&\qquad value : Values]\\
Message &\triangleq P1aMessage \cup P1bMessage \cup P2aMessage \cup P2bMessage
\end{aligned}$$

ASSUME $PaxosAssume \triangleq$

$$\begin{aligned}
&\land\ IsFiniteSet(Replicas)\\
&\land\ any \notin Values \cup \{none\}\\
&\land\ none \notin Values \cup \{any\}\\
&\land\ Ballots \subseteq Nat \land 0 \in Ballots\\
&\land\ \forall q \in Quorums : q \subseteq Replicas\\
&\land\ \forall q \in Quorums : Cardinality(Replicas) \div 2 < Cardinality(q)\\
&\land\ \forall q, r \in Quorums : q \cap r \neq \{\}
\end{aligned}$$

$p1aMessages \triangleq \{m \in messages : m.type = \text{"P1a"}\}$ Set of all $P1a$ messages sent.
$p1bMessages \triangleq \{m \in messages : m.type = \text{"P1b"}\}$ Set of all $P1b$ messages sent.
$p2aMessages \triangleq \{m \in messages : m.type = \text{"P2a"}\}$ Set of all $P2a$ messages sent.
$p2bMessages \triangleq \{m \in messages : m.type = \text{"P2b"}\}$ Set of all $P2b$ messages sent.

$ForcedValue(M) \triangleq (\text{CHOOSE } m \in M : \forall n \in M : n.maxVBallot \leq m.maxVBallot).maxValue$

$SendMessage(m) \triangleq messages' = messages \cup \{m\}$

Phase $1a$:

A proposer creates a message, which we call a "Prepare", identified with a ballot number b. Note that b is not the value to be proposed and maybe agreed on, but just a number which uniquely identifies this initial message by the proposer (to be sent to the acceptors).

While ballot number b must be greater than any ballot number used in any of the previous Prepare messages by this proposer, since the system is asynchronous and messages may be delayed and arrive out-of-order, there is no need to explicitly model this.

Then, it sends the Prepare message containing b to at least a quorum of acceptors. Note that the Prepare message only contains the ballot number b (that is, it does not have to contain the proposed value).

A Proposer should not initiate *Paxos* if it cannot communicate with at least a Quorum of Acceptors.

Some implementations may include the identity of the proposer, but that is omitted in this specification. Because while it is possible for multiple proposers to send a Prepare message with the same ballot number, only one of them can possibly receive a quorum of Promise replies. Thus, it is impossible for more than one proposer to have the same ballot number in Phase $2a$.

```
$PaxosPrepare \triangleq$
    $\land$ UNCHANGED $\langle decision,\ maxBallot,\ maxVBallot,\ maxValue \rangle$
    $\land \exists\, b \in Ballots \setminus \{0\}$ :
        $SendMessage([type \mapsto$ "P1a",
                      $ballot \mapsto b])$
```

Phase $1b$:

Any of the acceptors waits for a Prepare message from any of the proposers. If an acceptor receives a Prepare message, the acceptor must look at the ballot number b of the just received Prepare message.

There are two cases:

1. If b is higher than every previous proposal number received, from any of the proposers, by the acceptor, then the acceptor must return a message, which we call a "Promise", to the proposer, to ignore all future proposals having a ballot less than b. If the acceptor accepted a proposal at some point in the past, it must include the previous proposal number and the corresponding accepted value in its response to the proposer.

2. Otherwise the acceptor can ignore the received proposal. It does not have to answer in this case for *Paxos* to work.

```
$PaxosPromise \triangleq$
    $\land$ UNCHANGED $\langle decision,\ maxVBallot,\ maxValue \rangle$
    $\land \exists\, a \in Replicas,\ m \in p1aMessages$ :
        $\land\ maxBallot[a] < m.ballot$
        $\land\ maxBallot' = [maxBallot$ EXCEPT $![a] = m.ballot]$
        $\land\ SendMessage([type \mapsto$ "P1b",
                            $ballot \mapsto m.ballot,$
                            $acceptor \mapsto a,$
                            $maxVBallot \mapsto maxVBallot[a],$
                            $maxValue \mapsto maxValue[a]])$
```

Phase $2a$:

If a proposer receives Promises from a quorum of acceptors, it needs to set a value v to its proposal. If any acceptors had previously accepted any proposal, then they'll have sent their values to the proposer, who now must set the value of its proposal, v, to the value associated with the highest proposal ballot reported by the acceptors, let's call it z. If none of the acceptors had accepted a proposal up to this point, then the proposer may choose the value it originally wanted to propose, say x.

The proposer sends an Accept message, $(b,\ v)$, to a quorum of acceptors with the chosen value for its proposal, v, and the ballot number b (which is the same as the number contained in the Prepare message previously sent to the acceptors). So, the Accept message is either $(b,\ v = z)$ or, in case none of the Acceptors previously accepted a value, $(b,\ v = x)$.

This Accept message should be interpreted as a "request", as in "Accept this proposal, please!".

```
$PaxosAccept \triangleq$
    $\land$ UNCHANGED $\langle decision, maxBallot, maxVBallot, maxValue \rangle$
    $\land \exists\, b \in Ballots, q \in Quorums, v \in Values :$
        $\land \forall\, m \in p2aMessages : \neg(m.ballot = b)$
        $\land$ LET $M \triangleq \{m \in p1bMessages : m.ballot = b \land m.acceptor \in q\}$
          IN   $\land \forall\, a \in q : \exists\, m \in M : m.acceptor = a$
               $\land\ \lor \forall\, m \in M : m.maxValue = none$
                  $\lor\ v = ForcedValue(M)$
               $\land\ SendMessage([type \mapsto$ "P2a",
                                  $ballot \mapsto b,$
                                  $value \mapsto v])$
```

Phase $2b$:

If an acceptor receives an Accept message, $(b,\ v)$, from a proposer, it must accept it if and only if it has not already promised (in Phase $1b$ of the *Paxos* protocol) to only consider proposals having a ballot greater than b.

If the acceptor has not already promised (in Phase $1b$) to only consider proposals having a ballot greater than b, it should register the value v (of the just received Accept message) as the accepted value (of the protocol), and send an Accepted message to the proposer and every acceptor.

Else, it can ignore the Accept message or request.

```
$PaxosAccepted \triangleq$
    $\land$ UNCHANGED $\langle decision \rangle$
    $\land \exists\, a \in Replicas, m \in p2aMessages :$
        $\land\ m.value \in Values$
        $\land\ maxBallot[a] \leq m.ballot$
        $\land\ maxBallot' = [maxBallot$ EXCEPT $![a] = m.ballot]$
        $\land\ maxVBallot' = [maxVBallot$ EXCEPT $![a] = m.ballot]$
        $\land\ maxValue' = [maxValue$ EXCEPT $![a] = m.value]$
        $\land\ SendMessage([type \mapsto$ "P2b",
                         $ballot \mapsto m.ballot,$
                         $acceptor \mapsto a,$
                         $value \mapsto m.value])$
```

Consensus is achieved when a majority of acceptors accept the same ballot number (rather than the same value). Because each ballot is unique to a proposer and only one value may be proposed per ballot, all acceptors that accept the same ballot thereby accept the same value.

There is no need to model the variable decision for every acceptor. In this specification, the variable decision represents the decision of any acceptor, can its value may potentially be changed multiple times. Instead, we use the consistency safety property to proof that the decision for every acceptor is the same.

```
$PaxosDecide \triangleq$
    $\land$ UNCHANGED $\langle messages, maxBallot, maxVBallot, maxValue \rangle$
    $\land \exists\, b \in Ballots, q \in Quorums :$
        LET $M \triangleq \{m \in p2bMessages : m.ballot = b \land m.acceptor \in q\}$
        IN  $\land \forall\, a \in q : \exists\, m \in M : m.acceptor = a$
            $\land \exists\, m \in M : decision' = m.value$
```

```
$PaxosTypeOK \triangleq$ $\land$ $messages \subseteq Message$
                $\land$ $decision \in Values \cup \{none\}$
                $\land$ $maxBallot \in [Replicas \rightarrow Ballots]$
                $\land$ $maxVBallot \in [Replicas \rightarrow Ballots]$
                $\land$ $maxValue \in [Replicas \rightarrow Values \cup \{none\}]$
```

```
$PaxosInit \triangleq$ $\land$ $messages = \{\}$
              $\land$ $decision = none$
              $\land$ $maxBallot = [r \in Replicas \mapsto 0]$
              $\land$ $maxVBallot = [r \in Replicas \mapsto 0]$
              $\land$ $maxValue = [r \in Replicas \mapsto none]$
```

```
$PaxosNext \triangleq$ $\lor$ $PaxosPrepare$
              $\lor$ $PaxosPromise$
              $\lor$ $PaxosAccept$
              $\lor$ $PaxosAccepted$
              $\lor$ $PaxosDecide$
```

```
$PaxosSpec \triangleq$ $\land$ $PaxosInit$
              $\land$ $\Box[PaxosNext]_{\langle messages, decision, maxBallot, maxVBallot, maxValue \rangle}$
              $\land$ $\text{SF}_{\langle messages, decision, maxBallot, maxVBallot, maxValue \rangle}(PaxosDecide)$
```

Non-triviality safety property: Only proposed values can be learnt.

```
$PaxosNontriviality \triangleq$
    $\land$ $\lor$ $decision = none$
        $\lor$ $\exists\, m \in p2aMessages : m.value = decision$
    $\land$ $\forall\, m \in p1bMessages :$ $\land$ $m.maxValue \in Values \lor 0 = m.maxVBallot$
                                 $\land$ $m.maxValue = none \lor 0 < m.maxVBallot$
```

Consistency safety property: At most 1 value can be learnt.

$$PaxosConsistency \triangleq \Box[decision = none]_{\langle decision \rangle}$$

From *Wikipedia*:

Note that *Paxos* is not guaranteed to terminate, and thus does not have the liveness property. This is supported by the *Fischer* Lynch *Paterson* impossibility result (*FLP*) which states that a consistency protocol can only have two of safety, liveness, and fault tolerance.

As *Paxos*'s point is to ensure fault tolerance and it guarantees safety, it cannot also guarantee liveness.

$PaxosLiveness \triangleq \text{FALSE}$

Define symmetry for faster computations.

$PaxosSymmetry \triangleq Permutations(Values) \cup Permutations(Replicas)$