MODULE *TPaxos*

Copied from my junior.
(https://*github.com*/*Starydark*/PaxosStore-tla/blob/master/specification/*TPaxos.tla*)

EXTENDS *Integers*, *FiniteSets*

$Max(m,\ n) \triangleq$ IF $m > n$ THEN m ELSE n
$Injective(f) \triangleq \forall\, a,\ b \in$ DOMAIN $f : (a \neq b) \Rightarrow (f[a] \neq f[b])$

CONSTANTS *Participant*, the set of participants
Value the set of possible input values for *Participant* to propose

$None \triangleq$ CHOOSE $b : b \notin Value$

$Quorum \triangleq \{Q \in$ SUBSET $Participant : Cardinality(Q) * 2 > Cardinality(Participant)\}$

ASSUME $QuorumAssumption \triangleq \land \forall\, Q \in Quorum : Q \subseteq Participant$
$\land \forall\, Q1,\ Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$

$Ballot \triangleq Nat$

$MaxBallot \triangleq Cardinality(Ballot) - 1$

$PIndex \triangleq$ CHOOSE $f \in [Participant \to 1\,..\,Cardinality(Participant)] : Injective(f)$
$Bals(p) \triangleq \{b \in Ballot : b\%Cardinality(Participant) = PIndex[p] - 1\}$ allocate ballots for each $p \in Participant$

$State \triangleq [maxBal : Ballot \cup \{-1\},\ maxVBal : Ballot \cup \{-1\},\ maxVVal : Value \cup \{None\}]$

$InitState \triangleq [maxBal \mapsto -1,\ maxVBal \mapsto -1,\ maxVVal \mapsto None]$

For simplicity, in this specification, we choose to send the complete state of a participant each time. When receiving such a message, the participant processes only the "partial" state it needs.

$Message \triangleq [from : Participant,\ to :$ SUBSET $Participant,\ state : [Participant \to State]]$

VARIABLES *state*, *state*[*p*][*q*]: the state of $q \in Participant$ from the view of $p \in Participant$
msgs the set of messages that have been sent

$vars \triangleq \langle state,\ msgs \rangle$

$TypeOK \triangleq \land state \in [Participant \to [Participant \to State]]$
$\land msgs \subseteq Message$

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

$Init \triangleq \land state = [p \in Participant \mapsto [q \in Participant \mapsto InitState]]$
$\land msgs = \{\}$

$p \in Participant$ starts the prepare phase by issuing a ballot $b \in Ballot$.

$Prepare(p,\ b) \triangleq \land b \in Bals(p)$
$\land state[p][p].maxBal < b$

$\land\ state' = [state \text{ EXCEPT } ![p][p].maxBal = b]$
$\land\ Send([from \mapsto p,\ to \mapsto Participant \setminus \{p\},\ state \mapsto state'[p]])$

$q \in Participant$ updates its own state $state[q]$ according to the actual state pp of $p \in Participant$ extracted from a message $m \in Message$ it receives. This is called by $OnMessage(q)$. Note: pp is $m.state[p]$; it may not be equal to $state[p][p]$ at the time $UpdateState$ is called.

$$
\begin{array}{l}
UpdateState(q,\ p,\ pp) \triangleq \text{LET } maxB \triangleq Max(state[q][q].maxBal,\ pp.maxBal) \\
\qquad\qquad \text{IN} \quad state' = [state \text{ EXCEPT } ![q][p].maxBal = Max(@,\ pp.maxBal), \\
\qquad\qquad\qquad\qquad\qquad\qquad ![q][p].maxVBal = Max(@,\ pp.maxVBal), \\
\qquad\qquad\qquad\qquad\qquad\qquad ![q][p].maxVVal = \text{IF } state[q][p].maxVBal < pp.maxVB \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVVal \text{ ELSE } @, \\
\qquad\qquad\qquad\qquad\qquad\qquad ![q][q].maxBal = maxB, \qquad \text{make pron} \\
\qquad\qquad\qquad\qquad\qquad\qquad ![q][q].maxVBal = \text{IF } maxB \leq pp.maxVBal \qquad \text{accept} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVBal \text{ ELSE } @, \\
\qquad\qquad\qquad\qquad\qquad\qquad ![q][q].maxVVal = \text{IF } maxB \leq pp.maxVBal \qquad \text{accept} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVVal \text{ ELSE } @]
\end{array}
$$

$q \in Participant$ receives and processes a message in $Message$.

$$
\begin{array}{l}
OnMessage(q) \triangleq \exists\, m \in msgs : \\
\qquad \land\ q \in m.to \\
\qquad \land \text{LET } p \triangleq m.from \\
\qquad\quad \text{IN} \quad UpdateState(q,\ p,\ m.state[p]) \\
\qquad \land \text{LET } qm \triangleq [from \mapsto m.from,\ to \mapsto m.to \setminus \{q\},\ state \mapsto m.state] \qquad \text{remove } q \text{ from to} \\
\qquad\qquad\ \ nm \triangleq [from \mapsto q,\ to \mapsto \{m.from\},\ state \mapsto state'[q]] \qquad \text{new message to reply} \\
\qquad\quad \text{IN} \quad \text{IF } \lor\ m.state[q].maxBal < state'[q][q].maxBal \\
\qquad\qquad\qquad\ \ \lor\ m.state[q].maxVBal < state'[q][q].maxVBal \\
\qquad\qquad\quad \text{THEN } msgs' = (msgs \setminus \{m\}) \cup \{qm,\ nm\} \\
\qquad\qquad\quad \text{ELSE } msgs' = (msgs \setminus \{m\}) \cup \{qm\}
\end{array}
$$

$$
\begin{array}{l}
Accept(p,\ b,\ v) \triangleq \land\ b \in Bals(p) \\
\qquad\qquad\qquad\quad \land\ state[p][p].maxBal \leq b \qquad \text{corresponding the first conjunction in Voting} \\
\qquad\qquad\qquad\quad \land\ state[p][p].maxVBal \neq b \qquad \text{cooresponding the first conjunction in Voting} \\
\qquad\qquad\qquad\quad \land \exists\, Q \in Quorum : \\
\qquad\qquad\qquad\qquad \land \forall\, q \in Q \quad : state[p][q].maxBal = b \\
\qquad\qquad\qquad\quad \land\ \lor \forall\, q \in Participant : state[p][q].maxVBal = -1 \\
\qquad\qquad\qquad\qquad \lor \exists\, q \in Participant : \\
\qquad\qquad\qquad\qquad\qquad \land\ state[p][q].maxVVal = v \\
\qquad\qquad\qquad\qquad\qquad \land \forall\, r \in Participant : state[p][q].maxVBal \geq state[p][r].maxVBal \\
\qquad\qquad\qquad\quad \land\ state' = [state \text{ EXCEPT } ![p][p].maxVBal = b, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![p][p].maxVVal = v] \\
\qquad\qquad\qquad\quad \land\ Send([from \mapsto p,\ to \mapsto Participant \setminus \{p\},\ state \mapsto state'[p]])
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq \exists\, p \in Participant : \lor\ OnMessage(p) \\
\qquad\qquad\qquad\qquad\quad \lor \exists\, b \in Ballot : \lor\ Prepare(p,\ b) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \lor \exists\, v \in Value : Accept(p,\ b,\ v)
\end{array}
$$

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

$$ChosenP(p) \triangleq \{v \in Value : \exists\, b \in Ballot :$$
$$\exists\, Q \in Quorum : \forall\, q \in Q : \land\, state[p][q].maxVBal = b$$
$$\land\, state[p][q].maxVVal = v\}$$

$$chosen \triangleq \text{UNION}\ \{ChosenP(p) : p \in Participant\}$$

$$Consistency \triangleq Cardinality(chosen) \leq 1$$

THEOREM $Spec \Rightarrow \Box Consistency$

For checking *Liveness* $WF(A)$: if A ever becomes enabled, then an A step will eventually occur-even if A remains enabled for only a fraction of a nanosecond and is never again enabled. Liveness in *TPaxos*: like paxos, there should be a single-leader to prapre and accept.

$$\begin{aligned} LConstrain \triangleq\ & \land \exists\, p \in Participant : \\ & \land\, MaxBallot \in Bals(p) \\ & \land\, \mathrm{WF}_{vars}(Prepare(p,\, MaxBallot)) \\ & \land \forall\, v\ \in Value : \mathrm{WF}_{vars}(Accept(p,\, MaxBallot,\, v)) \\ & \land \exists\, Q \in Quorum : \\ & \quad \land\, p \in Q \\ & \quad \land \forall\, q \in Q : \mathrm{WF}_{vars}(OnMessage(q)) \end{aligned}$$

$$LSpec \triangleq Spec\ \land LConstrain$$

$$Liveness \triangleq \Diamond(chosen \neq \{\})$$

* Modification History
* Last modified *Mon Mar* 08 14:38:59 *CST* 2021 by Dell
* Created Sat *Mar* 06 18:00:00 *CST* 2021 by Dell