MODULE *Voting*

This is a high-level algorithm in which a set of processes cooperatively choose a value.

EXTENDS *Integers* paxos *PN Ballot*

ValueChosen

CONSTANT *Value*, The set of choosable values.

Acceptor, A set of processes that will choose a value.

Quorum The set of "quorums", where a quorum" is a "large enough" set of acceptors

Here are the assumptions we make about quorums.

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

THEOREM $QuorumNonEmpty \triangleq \forall Q \in Quorum : Q \neq \{\}$

Ballot is a set of "ballot numbers". For simplicity, we let it be the set of natural numbers. However, we write *Ballot* for that set to distinguish ballots from natural numbers used for other purposes.

$Ballot \triangleq Nat$ Ballotpn

In the algorithm, each acceptor can cast one or more votes, where each vote cast by an acceptor has the form $\langle b, v \rangle$ indicating that the acceptor has voted for value v in ballot b. A value is chosen if a quorum of acceptors have voted for it in the same ballot.

The algorithm's variables.

$votes[a]$ a $MaxBal[a]$a$Ballot(PN)$

VARIABLE *votes*, $votes[a]$ is the set of votes cast by acceptor a

maxBal $maxBal[a]$ is a ballot number. Acceptor a will cast further votes only in ballots numbered $\geq maxBal[a]$

acceptorproposal, $\times$ *Cartesian* product

The type-correctness invariant.

$TypeOK \triangleq \land votes \in [Acceptor \rightarrow \text{SUBSET } (Ballot \times Value)]$
$\land maxBal \in [Acceptor \rightarrow Ballot \cup \{-1\}]$

We now make a series of definitions an assert some simple theorems about those definitions that lead to the algorithm.

a *Ballot* b, value v ; $\langle b, v \rangle$, ordered tuples

$VotedFor(a, b, v) \triangleq \langle b, v \rangle \in votes[a]$

True iff acceptor a has voted for v in ballot b.

$Quorum\ Q(b, v)$

$ChosenAt(b, v) \triangleq \exists Q \in Quorum :$
$\forall a \in Q : VotedFor(a, b, v)$

True iff a quorum of acceptors have all voted for v in ballot b.

chosen$v$ $Ballot$ $b$chosen

$chosen \triangleq \{v \in Value : \exists\, b \in Ballot : ChosenAt(b, v)\}$

The set of values that have been chosen.

$Ballot\ bv$

$DidNotVoteAt(a, b) \triangleq \forall\, v \in Value : \neg VotedFor(a, b, v)$

$\geq$? $Ballot$paxosdistinct setserverPN

v, b

$DidNotVoteAt(a, b) bv(b, v)$ $TODO$: paxosPN

$$CannotVoteAt(a, b) \triangleq \land\ maxBal[a] > b$$
$$\land\ DidNotVoteAt(a, b)$$

Because acceptor a will not cast any more votes in a ballot numbered $< maxBal[a]$, this implies that a has not and will never cast a vote in ballot b.

$vw\,Ballot\ b$ choose$Quorum(b, v) b$

quorum

$NoneOtherChoosableAt(b, v) \triangleq$

$\quad \exists\, Q \in Quorum :$

$\quad\quad \forall\, a \in Q : VotedFor(a, b, v) \lor CannotVoteAt(a, b)$

If this is true, then $ChosenAt(b, w)$ is not and can never become true for any $w \neq v$.

vbballot chosenballotPhase 2

$< b, v >$ $Ballot\,v$ $P2b$?

$SafeAt(b, v) \triangleq \forall\, c \in 0 \,..\, (b - 1) : NoneOtherChoosableAt(c, v)$

If this is true, then no value other than v has been or can ever be chosen in any ballot numbered less than b.

ballot 0

THEOREM $AllSafeAtZero \triangleq \forall\, v \in Value : SafeAt(0, v)$

THEOREM $ChoosableThm \triangleq$

$\quad \forall\, b \in Ballot, v \in Value :$

$\quad\quad ChosenAt(b, v) \Rightarrow NoneOtherChoosableAt(b, v)$

$CannotVoteAt$ chosen $VotedFor(a, b, v) SafeAt(b, v)$

$VotesSafe \triangleq \forall\, a \in Acceptor, b \in Ballot, v \in Value :$

$\quad VotedFor(a, b, v) \Rightarrow SafeAt(b, v)$

a$b$value

$ToDo$: $a1a1b$value?

$OneVote \triangleq \forall\, a \in Acceptor, b \in Ballot, v, w \in Value :$

$\quad VotedFor(a, b, v) \land VotedFor(a, b, w) \Rightarrow (v = w)$

$OneValuePerBallot \triangleq$

$\quad \forall\, a1, a2 \in Acceptor, b \in Ballot, v1, v2 \in Value :$

$\quad\quad VotedFor(a1, b, v1) \land VotedFor(a2, b, v2) \Rightarrow (v1 = v2)$

THEOREM $OneValuePerBallot \Rightarrow OneVote$

Consensus

THEOREM $VotesSafeImpliesConsistency \triangleq$

$$\begin{aligned}
&\land\ TypeOK \\
&\land\ VotesSafe \\
&\land\ OneVote \\
&\Rightarrow\ \lor\ chosen = \{\} \\
&\qquad \lor\ \exists\, v \in Value : chosen = \{v\}
\end{aligned}$$

$Q,\ b,\ v$

$$\begin{aligned}
&ShowsSafeAt(Q,\ b,\ v) \triangleq \\
&\quad \land\ \forall\, a \in Q : maxBal[a] \geq b \\
&\quad \land\ \exists\, c \in\ -1 \,..\, (b-1) : \\
&\qquad \land\ (c \neq -1) \Rightarrow \exists\, a \in Q : VotedFor(a,\ c,\ v) \\
&\qquad \land\ \forall\, d \in (c+1) \,..\, (b-1),\ a \in Q : DidNotVoteAt(a,\ d)
\end{aligned}$$

THEOREM $ShowsSafety \triangleq$

$$\begin{aligned}
&TypeOK \land VotesSafe \land OneValuePerBallot \Rightarrow \\
&\quad \forall\, Q \in Quorum,\ b \in Ballot,\ v \in Value : \\
&\qquad ShowsSafeAt(Q,\ b,\ v) \Rightarrow SafeAt(b,\ v)
\end{aligned}$$

We now write the specification. The initial condition is straightforward.

servervotes$maxBal$ -1

$$\begin{aligned}
Init \triangleq\ &\land\ votes = [a \in Acceptor \mapsto \{\}] \\
&\land\ maxBal = [a \in Acceptor \mapsto -1]
\end{aligned}$$

Next are the actions that make up the next-state action.

An acceptor a is allowed to increase $maxBal[a]$ to a ballot number b at any time.

$maxBal$server

$$\begin{aligned}
&IncreaseMaxBal(a,\ b) \triangleq \\
&\quad \land\ b > maxBal[a] \\
&\quad \land\ maxBal' = [maxBal\ \text{EXCEPT}\ ![a] = b] \\
&\quad \land\ \text{UNCHANGED}\ votes
\end{aligned}$$

Next is the action in which acceptor a votes for v in ballot b. The first four conjuncts re enabling conditions. The first maintains the requirement that the acceptor cannot cast a vote in a ballot less than $maxBal[a]$. The next two conjuncts maintain the invariance of $OneValuePerBallot$. The fourth conjunct maintains the invariance of $VotesSafe$.

$VoteFor(a,\ b,\ v) \triangleq$

$\land\ maxBal[a] \leq b$ — a$Ballot \leq b$

$\land\ \forall\, vt \in votes[a] : vt[1] \neq b$ — a ballot b

$\land\ \forall\, c \in Acceptor \setminus \{a\} :$ — aserver$Ballot$ $b$value b

$$\forall\, vt \in votes[c] : (vt[1] = b) \Rightarrow (vt[2] = v)$$
$$\land\ \exists\, Q \in Quorum : ShowsSafeAt(Q, b, v)$$
$$\land\ votes' = [votes \text{ EXCEPT } ![a] = @ \cup \{\langle b, v\rangle\}]$$
$$\land\ maxBal' = [maxBal \text{ EXCEPT } ![a] = b]$$

The next-state action and the invariant.

Nextserver*Ballot*Votephase phase 2?

$$Next \triangleq \exists\, a \in Acceptor,\ b \in Ballot : \lor\ IncreaseMaxBal(a, b) \lor \exists\, v \in Value : VoteFor(a, b, v)$$

$$Spec \triangleq Init \land \Box[Next]_{\langle votes,\, maxBal\rangle}$$

$$Inv \triangleq TypeOK \land VotesSafe \land OneValuePerBallot$$

THEOREM $Invariance \triangleq Spec \Rightarrow \Box Inv$

The following statement instantiates module *Consensus* with the constant Value of this module substituted for the constant *Value* of module Consensus, and the state function *chosen* defined in this module substituted for the variable *chosen* of module *Value*. More precisely, for each defined identifier *id* of module *Value*, this statement defines $C!id$ to equal the value of *id* under these substitutions.

$C \triangleq$ INSTANCE *Consensus*

Consensuschosenvoting chosen. chosenvotingconsensus

TLA+ proofhttp://*lamport.azurewebsites.net*/pubs/keappa08-*web.pdf*

THEOREM $Spec \Rightarrow C!Spec$

$\langle 1\rangle 1.$ $Inv \land Init \Rightarrow C!Init$

$\langle 1\rangle 2.$ $Inv \land [Next]_{\langle votes,\, maxBal\rangle} \Rightarrow [C!Next]_{chosen}$

$\langle 1\rangle 3.$ QED

  $\langle 2\rangle 1.$ $\Box Inv \land \Box[Next]_{\langle votes,\, maxBal\rangle} \Rightarrow \Box[C!Next]_{chosen}$

    BY $\langle 1\rangle 2$ and temporal reasoning

  $\langle 2\rangle 2.$ $\Box Inv \land Spec \Rightarrow C!Spec$

    BY $\langle 2\rangle 1$, $\langle 1\rangle 1$

  $\langle 2\rangle 3.$ QED

    BY $\langle 2\rangle 2$, *Invariance*