MODULE *Voting*

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

EXTENDS *Integers*

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.

$Ballot \triangleq Nat$

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.

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 $\geq maxBal[a]$

The type-correctness invariant.

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

为什么直接说vote，不说phase 1？因为未形成决议的phase1，需要保证的比较简单

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

$VotedFor(a, b, v) \triangleq \langle b, v \rangle \in votes[a]$ vote是带了b和v的，已经有v。不是phase 1

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

$ChosenAt(b, v) \triangleq \exists Q \in Quorum :$
$\forall a \in Q : VotedFor(a, b, v)$ 判断出现了chosen，即形成决议

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

$chosen \triangleq \{v \in Value : \exists b \in Ballot : ChosenAt(b, v)\}$ 一系列已经chosen的value

The set of values that have been chosen.

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

Server a,没有为ballot b透过票。也就是被更大的ballot跳过了

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

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.

如果别人要求用ballot c，choose x，那么必须有个quorum Q2同意。但是已经存在quorum Q1，选择了(b,v)，或者maxb>c
则不可能存在Q2,凑不够，因为Q1和Q2必然相交。

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

如果都vote了(b,v)，那么就不能换v了。还有可能，就是部分server的maxBal > b，也不可能在接受v以外的值。
所以肯定不会接受(b,w)，w/=v。 注意，这是针对ballot b来说的。换更大的，不一定如此

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

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

注意c的取值范围，不含b
为什么小于b的都没有choose成功？因为针对的是单paxos，choose就结束了

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

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

THEOREM $ChoosableThm \triangleq$

$$\forall\, b \in Ballot,\, v \in Value : \\ ChosenAt(b,\, v) \Rightarrow NoneOtherChoosableAt(b,\, v)$$

这是最关键的保证：一旦形成决议，b不可能再选择别的值

$$VotesSafe \triangleq \forall\, a \in Acceptor,\, b \in Ballot,\, v \in Value : \\ VotedFor(a,\, b,\, v) \Rightarrow SafeAt(b,\, v)$$

注意SafeAt的参数b

$$OneVote \triangleq \forall\, a \in Acceptor,\, b \in Ballot,\, v,\, w \in Value : \\ VotedFor(a,\, b,\, v) \land VotedFor(a,\, b,\, w) \Rightarrow (v = w)$$

对于任意指定server a，如果两次为同一个b vote，那么value一定是一个

$$OneValuePerBallot \triangleq \\ \forall\, a1,\, a2 \in Acceptor,\, b \in Ballot,\, v1,\, v2 \in Value : \\ VotedFor(a1,\, b,\, v1) \land VotedFor(a2,\, b,\, v2) \Rightarrow (v1 = v2)$$

与上一行不同，这里考虑的是同一b，不同a之间的关系。如果vote了，那么必须是相同值。

THEOREM $OneValuePerBallot \Rightarrow OneVote$

THEOREM $VotesSafeImpliesConsistency \triangleq$

$$\land\, TypeOK \\ \land\, VotesSafe \\ \land\, OneVote \\ \Rightarrow\, \lor\, chosen = \{\} \\ \lor\, \exists\, v \in Value : chosen = \{v\}$$

这个集合，就一个元素了？
==>因为是单实例，实际上就一个元素。只有ballot number，没有instance number

从quorum Q看来，(b,v)是安全的

$$ShowsSafeAt(Q,\, b,\, v) \triangleq \\ \land \forall\, a \in Q : maxBal[a] \geq b \\ \land \exists\, c \in -1\,..\,(b-1) : \\ \land\, (c \neq -1) \Rightarrow \exists\, a \in Q : VotedFor(a,\, c,\, v) \\ \land \forall\, d \in (c+1)\,..\,(b-1),\, a \in Q : DidNotVoteAt(a,\, d)$$

注意，这个是用来蕴含其他的，而不是被蕴含
都大于b了
存在一个c < b，a在ballot=c时，选了v，其次后再也没有投票过。
对于非-1的c，一定有a，选择了v
大于c的，任意d，以及任意a属于Q，都没有为d投票

THEOREM $ShowsSafety \triangleq$

$$TypeOK \land VotesSafe \land OneValuePerBallot \Rightarrow$$

$$\forall\, Q \in Quorum,\ b \in Ballot,\ v \in Value :$$
$$ShowsSafeAt(Q,\ b,\ v) \Rightarrow SafeAt(b,\ v)$$

任意一个Q，如果从它的角度，b,v安全，则整个(不仅仅是Q)，对于b,v都是安全的。

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

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

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.

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

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 $VotesSafe$.

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

The next-state action and the invariant.

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

为什么只有acceptor？没有纳入Proposal的过程？ 但是paxos有？

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

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

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.

$$C \triangleq \text{INSTANCE } Consensus$$

THEOREM $Spec \Rightarrow C!Spec$
$\langle 1 \rangle 1.\ Inv \wedge Init \Rightarrow C!Init$
$\langle 1 \rangle 2.\ Inv \wedge [Next]_{\langle votes,\, maxBal \rangle} \Rightarrow [C!Next]_{chosen}$
$\langle 1 \rangle 3.$ QED
  $\langle 2 \rangle 1. \Box Inv \wedge \Box[Next]_{\langle votes,\, maxBal \rangle} \Rightarrow \Box[C!Next]_{chosen}$
    BY $\langle 1 \rangle 2$ and temporal reasoning
  $\langle 2 \rangle 2. \Box Inv \wedge Spec \Rightarrow C!Spec$
    BY $\langle 2 \rangle 1$, $\langle 1 \rangle 1$
  $\langle 2 \rangle 3.$ QED
    BY $\langle 2 \rangle 2$, *Invariance*