```
1  ---------- MODULE MCVoting ----------
2  EXTENDS Voting, TLC

4  CONSTANTS a1, a2, a3   acceptors
5  CONSTANTS v1, v2       Values

7  MCAcceptor ≜ {a1, a2, a3}
8  MCValue ≜ {v1, v2}
9  MCQuorum ≜ {{a1, a2}, {a1, a3}, {a2, a3}}
10 MCBallot ≜ 0 .. 1
11 MCSymmetry ≜ Permutations(MCAcceptor) ∪ Permutations(MCValue)
```

The various formulas given to *TLC* through the configuration file must consist of single identifiers. Thus, to get *TLC* to check that the specification satisfies (implements) $C!Spec$, we cannot put

PROPERTY $C!Spec$

in the configuration file. We therefore define *ConsensusSpecBar* to equal it and put

PROPERTY *ConsensusSpecBar*

in the configuration file.

```
27 ConsensusSpecBar ≜ C!Spec
```

The following assumption checks theorem *QuorumNonEmpty*

```
32 ASSUME QuorumNonEmpty! :

34 MCInit ≜ TypeOK
```

Checking that *MCInv* is an invariant of *MCSpec* checks the correctness of theorems *AllSafeAtZero* through *ShowsSafety*.

```
40 MCSpec ≜ TypeOK ∧ □[FALSE]⟨votes, maxBal⟩
41 MCInv ≜ ∧ AllSafeAtZero! :
42         ∧ ChoosableThm! :
43         ∧ OneValuePerBallot ⇒ OneVote
44         ∧ VotesSafeImpliesConsistency! :
45         ∧ ShowsSafety! :
```

Checking that *Inv* is an invariant of *MCSpecI* checks that *Inv* is an inductive invariant–that is, it checks

THEOREM $Inv \land [Next]\_\langle votes, maxBal\rangle \Rightarrow Inv'$

```
53 MCSpecI ≜ Inv ∧ □[Next]⟨votes, maxBal⟩

55 ==========
```