```
──────────────── MODULE leaderevote ────────────────

EXTENDS Naturals, TLC, Sequences, FiniteSets, Integers, ExternalSeqRecordParser3
 Sequence to Set
RECURSIVE Seq2Set(_)
Seq2Set(S) ≜
    IF S = ⟨⟩ THEN {}
     ELSE
        LET i ≜ Head(S)
        IN   {i} ∪ Seq2Set(Tail(S))


 判断本轮选票是否存在state为FOLLOWING的选票
decideStateExisted(S, state) ≜ ∃x ∈ S :
                                      ∧ x.state = state


 判断本轮选票state全部为LOOKING


VARIABLES evoteSeq, offset, evotecollection
vars ≜ ⟨evoteSeq, evotecollection, offset⟩
Trace ≜ ExSeqRcdParser3("D:\\00001code\\runtimemodel\\zookeeper_environment\\leaderelection.log")

Init  ≜ ∧ offset = 1
          ∧ evoteSeq = ⟨⟩
          ∧ evotecollection = {}

term ≜ ∧ offset > Len(Trace)
          ∧ UNCHANGED vars

 过滤出来源于本节点的选票
```

```
$selectvoteFromNodeSelf(S) \triangleq$ CHOOSE $x \in S$ :
                                          $\land\ x.myId = x.from$
                                          $\land\ x.myState =$ "LOOKING"
                                          $\land\ x.state =$ "LOOKING"
$Rule1 \triangleq\ \land\ offset \leq Len(ExSeqRcdParser3($"./leaderelection.log"$))$
           $\land\ offset' = offset + 1$
           $\land\ evoteSeq' = ExSeqRcdParser3($"./leaderelection.log"$)[offset]$
           $\land\ evotecollection' = Seq2Set(evoteSeq')$
           $\land\ Assert(selectvoteFromNodeSelf(evotecollection').proposedLeader =$
                $selectvoteFromNodeSelf(evotecollection').myId,$
                "The node first vote itself as the leader.")


过滤本轮选票中最大的的$electionEpoch$选票
$selectMaxelectionEpoch(S) \triangleq \{x \in S :$
                              $\forall\, y \in S :$
                                $y.electionEpoch \leq x.electionEpoch\}$
过滤事务zxid最大的选票
$selectVoteByZxid(S) \triangleq \{x \in S :$
                         $\forall\, y \in S :$
                           $\lor\ y.proposedZxidHigh < x.proposedZxidHigh$
                           $\lor\ ((y.proposedZxidHigh = x.proposedZxidHigh)$
                              $\land\ (y.proposedZxidLow \leq x.proposedZxidLow)$
                             $)\}$
过滤事务$myId$最大的选票
$selectVoteByMyid(S) \triangleq \{x \in S :$
                        $\forall\, y \in S :$
                          $y.from \leq x.from\}$

判断选票状态是否全部为"LOOKING"
$decideStateAllIsLOOKING(S) \triangleq \forall\, x \in S :$
                                    $x.state =$ "LOOKING"
```

```
判断准leader与选举结束后的leader是否相同
$decideProposedLeaderEqualEndvote(S) \triangleq \forall x \in S :$
                                        $\land x.proposedLeader = x.endvote$
过滤指定state的选票
$selectVoteByState(S, state) \triangleq \{x \in S :$
                                $x.state = state\}$
$Rule3 \triangleq$ $\land offset \leq Len(ExSeqRcdParser3($“./leaderelection.log”$))$
          $\land offset' = offset + 1$
          $\land evoteSeq' = ExSeqRcdParser3($“./leaderelection.log”$)[offset]$
          $\land evotecollection' = Seq2Set(evoteSeq')$
          $\land$ IF $(decideStateAllIsLOOKING(selectMaxelectionEpoch(evotecollection')) = $ TRUE$)$
            THEN
             $\land Assert(decideProposedLeaderEqualEndvote(selectVoteByMyid(selectVoteByZxid($
                      $selectMaxelectionEpoch(evotecollection'))))) = $ TRUE,
               “Determine whether the end vote is consistent with the vote algorithm”$)$
            ELSE
             新节点加入集群
             $\land Assert(decideProposedLeaderEqualEndvote(selectVoteByState($
                      $selectMaxelectionEpoch(evotecollection'),$ “FOLLOWING”$)) = $ TRUE,
               “node joins the cluster”$)$
             $\land Assert(decideProposedLeaderEqualEndvote(selectVoteByState($
                     $selectMaxelectionEpoch(evotecollection'),$ “LEADING”$)) = $ TRUE,
               “node joins the cluster”$)$


$election \triangleq$ $\land offset \leq Len(Trace)$
             $\land offset' = offset + 1$
             $\land evoteSeq' = Trace[offset]$
             $\land evotecollection' = Seq2Set(evoteSeq')$
             $\land$ IF $((decideStateExisted(selectMaxelectionEpoch(evotecollection'),$ “FOLLOWING”$)) = $ TRUE$)$
               THEN $\land Assert(decideProposedLeaderEqualEndvote(selectVoteByState(selectMaxelectionEpoch$
               ELSE
                $\land$ TRUE
                ∧ IF ((decideStateExisted(selectMaxelectionEpoch(evotecollection′), "LEADING")) = TI
                    THEN  ∧ Assert(decideProposedLeaderEqualEndvote(selectVoteByState(selectMaxele
                    ELSE
                      ∧ TRUE
                ∧ Assert(selectvoteFromNodeSelf(evotecollection′).proposedLeader = selectvoteFromNo
                ∧ IF (decideStateAllIsLOOKING(selectMaxelectionEpoch(evotecollection′)) = TRUE)
                    THEN
                     ∧ Assert(decideProposedLeaderEqualEndvote(selectVoteByMyid(selectVoteByZxid(
                    ELSE
                     ∧  TRUE

Next ≜  ∨ election
        ∨ Rule3

Spec ≜ Init ∧ □[Next]_vars
```

* Modification History
* Last modified *Wed Mar* 16 19:25:10 *CST* 2022 by 10222803
* Created *Thu Feb* 24 16:12:54 *CST* 2022 by 10222803