```
┌──────────────── MODULE Proposal ────────────────┐
EXTENDS Configuration, Mastership

INSTANCE Naturals

INSTANCE FiniteSets

LOCAL INSTANCE TLC

├─────────────────────────────────────────────────┤

  Transaction type constants
CONSTANTS
   ProposalChange,
   ProposalRollback

  Phase constants
CONSTANTS
   ProposalValidate,
   ProposalCommit,
   ProposalApply,
   ProposalAbort

  Status constants
CONSTANTS
   ProposalInProgress,
   ProposalComplete,
   ProposalFailed

CONSTANT TraceProposal

  A record of per-target proposals
VARIABLE proposal

├─────────────────────────────────────────────────┤

LOCAL InitState ≜ [
   proposals      ↦ proposal,
   configurations ↦ configuration,
   targets         ↦ target,
   masterships     ↦ mastership,
   node           ↦ node]

LOCAL NextState ≜ [
   proposals      ↦ proposal′,
   configurations ↦ configuration′,
   targets         ↦ target′,
   masterships     ↦ mastership′,
    $node \qquad \mapsto node']$

LOCAL $Trace \triangleq$ INSTANCE $Trace$ WITH
    $Module \leftarrow$ "Proposals",
    $InitState \leftarrow InitState$,
    $NextState \leftarrow NextState$,
    $Enabled \leftarrow TraceProposal$

------------------------------------------------------------

$IsCommitted(i) \triangleq$
    $i \in$ DOMAIN $proposal \Rightarrow$
      CASE $proposal[i].phase = ProposalValidate \rightarrow$
               $proposal[i].state = ProposalFailed$
        $\Box$ $proposal[i].phase = ProposalCommit \rightarrow$
               $proposal[i].state \in \{ProposalComplete, ProposalFailed\}$
        $\Box$ OTHER $\rightarrow$ TRUE

$IsApplied(i) \triangleq$
    $i \in$ DOMAIN $proposal \Rightarrow$
      CASE $proposal[i].phase \in \{ProposalValidate, ProposalCommit\} \rightarrow$
               $proposal[i].state = ProposalFailed$
        $\Box$ $proposal[i].phase = ProposalCommit \rightarrow$
               $proposal[i].state \in \{ProposalComplete, ProposalFailed\}$
        $\Box$ OTHER $\rightarrow$ TRUE

 Reconcile a proposal
$ReconcileProposal(n, i) \triangleq$
     Only the master can process proposals for the target.
    $\land$ $mastership.master = n$
       While in the Validate phase, validate the proposed changes.
       If validation is successful, the proposal also records the changes
       required to roll back the proposal and the index to which to roll back.
    $\land$ $\lor$ $\land$ $proposal[i].phase = ProposalValidate$
          Validate proposals once the prior proposal has been committed.
         $\land$ $IsCommitted(i - 1)$
         $\land$ $\lor$ $\land$ $proposal[i].state = ProposalInProgress$
                  For Change proposals validate the set of requested changes.
               $\land$ $\lor$ $\land$ $proposal[i].type = ProposalChange$
                     $\land$ LET $rollbackIndex \triangleq configuration.committed.index$
                             $rollbackValues \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
                                   IF $p \in$ DOMAIN $configuration.committed.values$ THEN
                                      $configuration.committed.values[p]$
                                    ELSE
                                      $[delete \mapsto$ TRUE$]]$
                  If all the change values are valid, record the changes required to roll
          back the proposal and the index to which the rollback changes
          will roll back the configuration.
        IN
            $\lor\ proposal' = [proposal\ \text{EXCEPT}\ ![i].rollback = [index \mapsto rollbackIndex,$
                                                                      $values \mapsto rollbackValues],$
                                                   $![i].state = ProposalComplete]$
             A proposal can fail validation at this point, in which case the proposal
             is marked failed.
            $\lor\ proposal' = [proposal\ \text{EXCEPT}\ ![i].state = ProposalFailed]$
      For Rollback proposals, validate the rollback changes which are
      proposal being rolled back.
      $\lor\ \land\ proposal[i].type = ProposalRollback$
          Rollbacks can only be performed on Change type proposals.
         $\land\ \lor\ \land\ proposal[proposal[i].rollback.index].type = ProposalChange$
                Only roll back the change if it's the lastest change made
                to the configuration based on the configuration index.
               $\land\ \lor\ \land\ configuration.committed.index = proposal[i].rollback.index$
                     $\land\ \text{LET}\ changeIndex \triangleq proposal[proposal[i].rollback.index].rollback.index$
                             $changeValues \triangleq proposal[proposal[i].rollback.index].rollback.values$
                             $rollbackValues \triangleq proposal[proposal[i].rollback.index].change.values$
                      Record the changes required to roll back the target proposal and the index to
                      which the configuration is being rolled back.
                      $\text{IN}\ \ \land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change = [index \mapsto changeIndex,$
                                                                              $values \mapsto changeValues],$
                                                         $![i].change = [index \mapsto proposal[i].chang$
                                                                              $values \mapsto changeValues],$
                                                         $![i].state = ProposalComplete]$
                If the Rollback target is not the most recent change to the configuration,
                fail validation for the proposal.
                 $\lor\ \land\ configuration.committed.index \neq proposal[i].rollback.index$
                     $\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].state = ProposalFailed]$
          If a Rollback proposal is attempting to roll back another Rollback,
          fail validation for the proposal.
           $\lor\ \land\ proposal[proposal[i].rollback.index].type = ProposalRollback$
               $\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].state = ProposalFailed]$
    $\land\ \text{UNCHANGED}\ \langle configuration, target\rangle$
  $\lor\ \land\ proposal[i].state = ProposalComplete$
    $\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].phase = ProposalCommit,$
                                     $![i].state = ProposalInProgress]$
    $\land\ \text{UNCHANGED}\ \langle configuration, target\rangle$
  When a proposal is marked failed, set the configuration index to the proposal
  index to unblock subsequent proposals.
  $\lor\ \land\ proposal[i].state = ProposalFailed$
    $\land\ configuration' = [configuration\ \text{EXCEPT}\ !.index = i]$
    $\land\ \text{UNCHANGED}\ \langle proposal, target\rangle$
```

```
While in the Commit state, commit the proposed changes to the configuration.
$\lor$ $\land$ $proposal[i].phase = ProposalCommit$
   $\land$ $\lor$ $\land$ $proposal[i].state = ProposalInProgress$
         Only commit the proposal if the prior proposal has already been committed.
         $\land$ $configuration.index = i - 1$
         $\land$ $configuration' = [configuration$ EXCEPT $!.committed.values = proposal[i].change.values,$
                                                   $!.committed.index = proposal[i].change.index,$
                                                   $!.index = i]$
         $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$
         $\land$ UNCHANGED $\langle target \rangle$
      $\lor$ $\land$ $proposal[i].state = ProposalComplete$
         $\land$ $proposal' = [proposal$ EXCEPT $![i].phase = ProposalApply,$
                                       $![i].state = ProposalInProgress]$
         $\land$ UNCHANGED $\langle configuration, target \rangle$
While in the Apply phase, apply the proposed changes to the target.
$\lor$ $\land$ $proposal[i].phase = ProposalApply$
      For the proposal to be applied, the node must be connected to a running target.
   $\land$ $\lor$ $\land$ $proposal[i].state = ProposalInProgress$
         $\land$ $node[n].connected$
         $\land$ $target.running$
         Verify the applied index is the previous proposal index to ensure
         changes are applied to the target in order.
         $\land$ $configuration.applied.index = i - 1$
         Verify the applied term is the current *mastership* term to ensure the
         configuration has been synchronized following restarts.
         $\land$ $configuration.applied.term = mastership.term$
         Model successful and failed target update requests.
         $\land$ $\lor$ $\land$ $target' = [target$ EXCEPT $!.values = proposal[i].change.values]$
               $\land$ $configuration' = [configuration$ EXCEPT
                                      $!.applied.index = i,$
                                      $!.applied.values = proposal[i].change.values$
                                          $@@ configuration.applied.values]$
               $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$
            If the proposal could not be applied, update the configuration's applied index
            and mark the proposal Failed.
            $\lor$ $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index = i]$
               $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$
               $\land$ UNCHANGED $\langle target \rangle$
$\lor$ $\land$ $proposal[i].phase = ProposalAbort$
   $\land$ $proposal[i].state = ProposalInProgress$
      If the configuration index is less than the proposal index, the proposal has
      not been committed, so it can be aborted without any additional changes required.
   $\land$ $\lor$ $\land$ $configuration.index = i - 1$
         $\land$ $configuration' = [configuration$ EXCEPT $!.index = i]$
         $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$
```

```
    $\land$ UNCHANGED $\langle target \rangle$
  If the proposal has already been committed to the configuration but hasn't yet
  been applied to the target, we need to finish applying the proposal and fail
  the abort attempt.
  $\lor$ $\land$ $configuration.index \geq i$
    $\land$ $configuration.applied.index = i - 1$
    $\land$ $configuration.applied.term = mastership.term$
    $\land$ $node[n].connected$
    $\land$ $target.running$
     Model successful and failed target update requests.
    $\land$ $\lor$ $\land$ $target' = [target$ EXCEPT $!.values = proposal[i].change.values]$
           $\land$ $configuration' = [configuration$ EXCEPT
                                  $!.applied.index = i,$
                                  $!.applied.values = proposal[i].change.values$
                                      $@@ configuration.applied.values]$
           $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$
        If the proposal could not be applied, update the configuration's applied index
        and mark the proposal Failed.
        $\lor$ $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index = i]$
           $\land$ $proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$
           $\land$ UNCHANGED $\langle target \rangle$
$\land$ UNCHANGED $\langle mastership, node \rangle$
```

Formal specification, constraints, and theorems.

```
$InitProposal \triangleq$
   $\land$ $proposal = [$
        $i \in \{\} \mapsto [$
          $phase \mapsto ProposalValidate,$
          $state \mapsto ProposalInProgress]]$
   $\land$ $Trace!Init$

$NextProposal \triangleq$
   $\lor$ $\exists n \in Node :$
        $\exists i \in$ DOMAIN $proposal :$
          $Trace!Step(ReconcileProposal(n, i), [node \mapsto n, index \mapsto i])$
```

---

* Modification History
* Last modified *Fri Apr* 21 19:15:11 *PDT* 2023 by *jhalterm*
* Last modified *Mon Feb* 21 01:24:12 *PST* 2022 by *jordanhalterman*
* Created Sun *Feb* 20 10:07:16 *PST* 2022 by *jordanhalterman*