```
──────────────── MODULE Proposal ────────────────
INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

INSTANCE TLC
─────────────────────────────────────────────────

  An empty constant
CONSTANT Nil

  Transaction type constants
CONSTANTS
   Change,
   Rollback

  Phase constants
CONSTANTS
   Initialize,
   Validate,
   Abort,
   Commit,
   Apply

Phase ≜
   {Initialize,
    Validate,
    Abort,
    Commit,
    Apply}

  Status constants
CONSTANTS
   InProgress,
   Complete,
   Failed

State ≜
   {InProgress,
    Complete,
    Failed}

  State constants
CONSTANTS
   Pending,
   Validated,
   Committed,
   Applied,
   Aborted

Status ≜
   {Pending,
    Validated,
    Committed,
    Applied,
    Aborted}

CONSTANTS
   Valid,
   Invalid

CONSTANTS
   Success,
   Failure

  The set of all nodes
CONSTANT Node
```

```
  A record of per-target proposals
VARIABLE proposal

  A record of per-target configurations
VARIABLE configuration

  A record of target states
VARIABLE target

  A record of target masterships
VARIABLE mastership

Test ≜ INSTANCE Test WITH
    File         ← "Proposal.log",
    CurrState ← [
        proposals      ↦ proposal,
        configuration ↦ configuration,
        mastership     ↦ mastership,
        target           ↦ target],
    SuccState ← [
        proposals      ↦ proposal′,
        configuration ↦ configuration′,
        mastership     ↦ mastership′,
    $target \quad\quad \mapsto target']$
```

```
 Reconcile a proposal
$ReconcileProposal(n, t, i) \triangleq$
   $\land \lor \land proposal[t][i].phase = Initialize$
           $\land proposal[t][i].state = InProgress$
           $\land proposal' = [proposal$ EXCEPT $![t] =$
                $[proposal[t]$ EXCEPT $![i].state = Complete,$
                                      $![i].dependency.index = configuration[t].proposal.index]]$
           $\land configuration' = [configuration$ EXCEPT $![t].proposal.index = i]$
           $\land$ UNCHANGED $\langle target \rangle$
        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.
      $\lor \land proposal[t][i].phase = Validate$
         $\land proposal[t][i].state = InProgress$
         $\land configuration[t].commit.index = proposal[t][i].dependency.index$
            For Change proposals validate the set of requested changes.
         $\land \lor \land proposal[t][i].type = Change$
               $\land$ LET $rollbackIndex \triangleq configuration[t].config.index$
                       $rollbackValues \triangleq [p \in$ DOMAIN $proposal[t][i].change.values \mapsto$
                                          IF $p \in$ DOMAIN $configuration[t].config.values$ THEN
                                             $configuration[t].config.values[p]$
                                           ELSE
                                             $[value \mapsto Nil,$
                                              $delete \mapsto$ TRUE$]]$
                   Model validation successes and failures with Valid and Invalid results.
                 IN  $\exists r \in \{Valid, Invalid\}:$
                        If the Change is Valid, record the changes required to roll
                        back the proposal and the index to which the rollback changes
                        will roll back the configuration.
                      $\lor \land r = Valid$
                         $\land proposal' = [proposal$ EXCEPT $![t] =$
                                  $[proposal[t]$ EXCEPT $![i].rollback.index = rollbackIndex,$
                                                        $![i].rollback.values = rollbackValues,$
                                                        $![i].state = Complete]]$
                      $\lor \land r = Invalid$
                         $\land proposal' = [proposal$ EXCEPT $![t] =$
                                  $[proposal[t]$ EXCEPT $![i].state = Failed]]$
            For Rollback proposals, validate the rollback changes which are
            proposal being rolled back.
           $\lor \land proposal[t][i].type = Rollback$
                 Rollbacks can only be performed on Change type proposals.
        $\land \lor \land proposal[t][proposal[t][i].rollback.index].type = Change$
                Only roll back the change if it's the lastest change made
                to the configuration based on the configuration index.
            $\land \lor \land configuration[t].config.index = proposal[t][i].rollback.index$
                    $\land$ LET $changeIndex \triangleq proposal[t][proposal[t][i].rollback.index].rollback.index$
                            $changeValues \triangleq proposal[t][proposal[t][i].rollback.index].rollback.values$
                            $rollbackValues \triangleq proposal[t][proposal[t][i].rollback.index].change.values$
                      IN $\exists r \in \{Valid, Invalid\}$ :
                            If the *Rollback* is *Valid*, record the changes required to
                            roll back the target proposal and the index to which the
                            configuration is being rolled back.
                            $\lor \land r = Valid$
                              $\land proposal' = [proposal$ EXCEPT $![t] =$
                                    $[proposal[t]$ EXCEPT $![i].change.index = changeIndex,$
                                                          $![i].change.values = changeValues,$
                                                          $![i].rollback.values = rollbackValues,$
                                                          $![i].state = Complete]]$
                            $\lor \land r = Invalid$
                              $\land proposal' = [proposal$ EXCEPT $![t] =$
                                                 $[proposal[t]$ EXCEPT $![i].state = Failed]]$
                If the *Rollback* target is not the most recent change to the configuration,
                fail validation for the proposal.
                $\lor \land configuration[t].config.index \neq proposal[t][i].rollback.index$
                  $\land proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Failed]]$
            If a *Rollback* proposal is attempting to roll back another *Rollback*,
            fail validation for the proposal.
            $\lor \land proposal[t][proposal[t][i].rollback.index].type = Rollback$
              $\land proposal' = [proposal$ EXCEPT $![t] =$
                    $[proposal[t]$ EXCEPT $![i].state = Failed]]$
    $\land$ UNCHANGED $\langle configuration, target \rangle$
  While in the *Commit* state, commit the proposed changes to the configuration.
  $\lor \land proposal[t][i].phase = Commit$
    $\land proposal[t][i].state = InProgress$
     Only commit the proposal if the prior proposal has already been committed.
    $\land configuration[t].commit.index = proposal[t][i].dependency.index$
    $\land configuration' = [configuration$ EXCEPT $![t].config.values = proposal[t][i].change.values,$
                                                 $![t].config.index = proposal[t][i].change.index,$
                                                 $![t].commit.index = i]$
    $\land proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Complete]]$
    $\land$ UNCHANGED $\langle target \rangle$
  While in the *Apply* phase, apply the proposed changes to the target.
  $\lor \land proposal[t][i].phase = Apply$
    $\land proposal[t][i].state = InProgress$
    $\land configuration[t].target.index = proposal[t][i].dependency.index$
    $\land configuration[t].target.term = mastership[t].term$
```

```
        $\land$ $mastership[t].master = n$
         Model successful and failed target update requests.
        $\land$ $\exists\, r \in \{Success,\ Failure\}$ :
           $\lor$ $\land$ $r = Success$
              $\land$ $target' = [target$ EXCEPT $![t] = proposal[t][i].change.values$ @@ $target[t]]$
              $\land$ $configuration' = [configuration$ EXCEPT
                                      $![t].target.index\ \ = i$,
                                      $![t].target.values\ = proposal[t][i].change.values$
                                         @@ $configuration[t].target.values]$
              $\land$ $proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Complete]]$
            If the proposal could not be applied, update the configuration's applied index
            and mark the proposal *Failed*.
           $\lor$ $\land$ $r = Failure$
              $\land$ $configuration' = [configuration$ EXCEPT $![t].target.index = i]$
              $\land$ $proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Failed]]$
              $\land$ UNCHANGED $\langle target \rangle$
     $\lor$ $\land$ $proposal[t][i].phase = Abort$
        $\land$ $proposal[t][i].state\ \ = InProgress$
           The *commit.index* will always be greater than or equal to the *target.index*.
           If only the *commit.index* matches the proposal's *dependency.index*, update
           the *commit.index* to enable commits of later proposals, but do not
           mark the *Abort* phase *Complete* until the *target.index* has been incremented.
        $\land$ $\lor$ $\land$ $configuration[t].commit.index = proposal[t][i].dependency.index$
             $\land$ $configuration' = [configuration$ EXCEPT $![t].commit.index = i]$
             $\land$ UNCHANGED $\langle proposal \rangle$
           If the configuration's *target.index* matches the proposal's *dependency.index*,
           update the *target.index* and mark the proposal *Complete* for the *Abort* phase.
           $\lor$ $\land$ $configuration[t].commit.index \geq i$
             $\land$ $configuration[t].target.index\ \ \ = proposal[t][i].dependency.index$
             $\land$ $configuration' = [configuration$ EXCEPT $![t].target.index = i]$
             $\land$ $proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Complete]]$
           If both the configuration's *commit.index* and *target.index* match the
           proposal's *dependency.index*, update the *commit.index* and *target.index*
           and mark the proposal *Complete* for the *Abort* phase.
           $\lor$ $\land$ $configuration[t].commit.index = proposal[t][i].dependency.index$
             $\land$ $configuration[t].target.index\ \ \ = proposal[t][i].dependency.index$
             $\land$ $configuration' = [configuration$ EXCEPT $![t].commit.index = i$,
                                                    $![t].target.index\ \ \ = i]$
             $\land$ $proposal' = [proposal$ EXCEPT $![t] = [proposal[t]$ EXCEPT $![i].state = Complete]]$
        $\land$ UNCHANGED $\langle target \rangle$
  $\land$ UNCHANGED $\langle mastership \rangle$
```