```
┌──────────────────────── MODULE Config ────────────────────────┐
INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

INSTANCE TLC

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

GenerateTestCases ≜ FALSE

Nil ≜ “<nil>”

Change ≜ “Change”
Rollback ≜ “Rollback”

Commit ≜ “Commit”
Apply ≜ “Apply”

Pending ≜ “Pending”
InProgress ≜ “InProgress”
Complete ≜ “Complete”
Aborted ≜ “Aborted”
Failed ≜ “Failed”
Done ≜ {Complete, Aborted, Failed}

Node ≜ {“node1”}

NumTransactions ≜ 4
NumTerms ≜ 2
NumConns ≜ 2

Path ≜ {“path1”}
Value ≜ {“value1”, “value2”}

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

  A transaction log. Transactions may either request a set
  of changes to a set of targets or rollback a prior change.
VARIABLE transaction

  A record of per-target proposals
VARIABLE proposal

  A record of per-target configurations
VARIABLE configuration

  A record of target masterships
```

```
VARIABLE mastership

  A record of node connections to the target
VARIABLE conn

  The target state
VARIABLE target

  A sequence of state changes used for model checking.
VARIABLE history

vars ≜ ⟨transaction, proposal, configuration, mastership, conn, target, history⟩
```

```
LOCAL Transaction ≜ INSTANCE Transaction

LOCAL Proposal ≜ INSTANCE Proposal

LOCAL Configuration ≜ INSTANCE Configuration

LOCAL Mastership ≜ INSTANCE Mastership

LOCAL Target ≜ INSTANCE Target
```

```
RequestChange(p, v) ≜
    ∧ Transaction!RequestChange(p, v)
    ∧ UNCHANGED ⟨mastership, conn, target, history⟩

RequestRollback(i) ≜
    ∧ Transaction!RequestRollback(i)
    ∧ UNCHANGED ⟨mastership, conn, target, history⟩

ReconcileTransaction(n, i) ≜
    ∧ i ∈ DOMAIN transaction
    ∧ Transaction!ReconcileTransaction(n, i)
    ∧ UNCHANGED ⟨mastership, conn, target, history⟩
    ∧ GenerateTestCases ⇒ Transaction!Test!Log([node ↦ n, index ↦ i])

ReconcileProposal(n, i) ≜
    ∧ i ∈ DOMAIN proposal
    ∧ Proposal!ReconcileProposal(n, i)
    ∧ UNCHANGED ⟨transaction⟩
    ∧ GenerateTestCases ⇒ Proposal!Test!Log([node ↦ n, index ↦ i])

ReconcileConfiguration(n) ≜
    ∧ Configuration!ReconcileConfiguration(n)
    ∧ UNCHANGED ⟨transaction, proposal, history⟩
    $\land\ GenerateTestCases \Rightarrow Configuration!Test!Log([node \mapsto n])$

$ReconcileMastership(n)\ \triangleq$
    $\land\ Mastership!ReconcileMastership(n)$
    $\land\ \text{UNCHANGED}\ \langle transaction, proposal, configuration, target, history\rangle$
    $\land\ GenerateTestCases \Rightarrow Mastership!Test!Log([node \mapsto n])$

$ConnectNode(n)\ \triangleq$
    $\land\ Target!Connect(n)$
    $\land\ \text{UNCHANGED}\ \langle transaction, proposal, configuration, mastership, history\rangle$

$DisconnectNode(n)\ \triangleq$
    $\land\ Target!Disconnect(n)$
    $\land\ \text{UNCHANGED}\ \langle transaction, proposal, configuration, mastership, history\rangle$

$StartTarget\ \triangleq$
    $\land\ Target!Start$
    $\land\ \text{UNCHANGED}\ \langle transaction, proposal, configuration, mastership, history\rangle$

$StopTarget\ \triangleq$
    $\land\ Target!Stop$
    $\land\ \text{UNCHANGED}\ \langle transaction, proposal, configuration, mastership, history\rangle$
```

Formal specification, constraints, and theorems.

```
$Init\ \triangleq$
    $\land\ transaction = [$
        $i \in \{\} \mapsto [$
          $type \quad\ \mapsto Change,$
          $index \quad \mapsto 0,$
          $values \ \ \mapsto [p \in \{\} \mapsto Nil],$
          $commit \mapsto Pending,$
          $apply \quad \mapsto Pending]]$
    $\land\ proposal = [$
        $i \in \{\} \mapsto [$
          $change \mapsto [$
             $phase \ \ \mapsto Nil,$
             $state \ \ \ \mapsto Nil,$
             $values \ \mapsto [$
                $p \in \{\} \ \ \mapsto [$
                   $index \mapsto 0,$
                   $value \ \mapsto Nil]]],$
          $rollback \mapsto [$
             $phase \ \ \mapsto Nil,$
             $state \ \ \ \mapsto Nil,$
             $values \ \mapsto [$
```

```
                 p   ∈ {}  ↦ [
                      index ↦ 0,
                      value ↦ Nil]]]]]
    ∧ configuration = [
          state   ↦ InProgress,
          term    ↦ 0,
          committed ↦ [
              index    ↦ 0,
              revision ↦ 0,
              values   ↦ [
                  p ∈ {}  ↦ [
                      index ↦ 0,
                      value ↦ Nil]]],
          applied ↦ [
              index    ↦ 0,
              revision ↦ 0,
              target   ↦ 0,
              values   ↦ [
                  p ∈ {}  ↦ [
                      index ↦ 0,
                      value ↦ Nil]]]]
    ∧ target = [
          id      ↦ 0,
          running ↦ FALSE,
          values  ↦ [
              p ∈ {}  ↦ [
                  index ↦ 0,
                  value ↦ Nil]]]
    ∧ mastership = [
          master ↦ Nil,
          term   ↦ 0,
          conn   ↦ 0]
    ∧ conn = [
          n ∈ Node ↦ [
              id        ↦ 0,
              connected ↦ FALSE]]
    ∧ history = ⟨⟩

Next ≜
    ∨ ∃ p ∈ Path, v ∈ Value :
        RequestChange(p, v)
    ∨ ∃ i ∈ DOMAIN transaction :
        RequestRollback(i)
    ∨ ∃ n ∈ Node :
        ∃ i ∈ DOMAIN transaction :
            $ReconcileTransaction(n, i)$
    $\lor \exists n \in Node :$
          $\exists i \in \text{DOMAIN } proposal :$
            $ReconcileProposal(n, i)$
    $\lor \exists n \in Node :$
          $ReconcileConfiguration(n)$
    $\lor \exists n \in Node :$
          $ReconcileMastership(n)$
    $\lor \exists n \in Node :$
          $\lor ConnectNode(n)$
          $\lor DisconnectNode(n)$
    $\lor StartTarget$
    $\lor StopTarget$

$Spec \triangleq$
    $\land Init$
    $\land \Box[Next]_{vars}$
    $\land \forall p \in Path, v \in Value :$
          $\text{WF}_{\langle transaction, proposal, configuration, mastership, target \rangle}(Transaction!RequestChange(p, v))$
    $\land \forall i \in 1 \,..\, NumTransactions : i \in \text{DOMAIN } transaction \Rightarrow$
          $\text{WF}_{\langle transaction, proposal, configuration, mastership, target \rangle}(Transaction!RequestRollback(i))$
    $\land \forall n \in Node, i \in 1 \,..\, NumTransactions :$
          $\text{WF}_{\langle transaction, proposal, configuration, mastership, target \rangle}(Transaction!ReconcileTransaction(n, i))$
    $\land \forall n \in Node, i \in 1 \,..\, NumTransactions :$
          $\text{WF}_{\langle proposal, configuration, mastership, conn, target, history \rangle}(Proposal!ReconcileProposal(n, i))$
    $\land \forall n \in Node :$
          $\text{WF}_{\langle configuration, mastership, conn, target \rangle}(Configuration!ReconcileConfiguration(n))$
    $\land \forall n \in Node :$
          $\text{WF}_{\langle mastership, conn \rangle}(Mastership!ReconcileMastership(n))$
    $\land \forall n \in Node :$
          $\text{WF}_{\langle conn, target \rangle}(Target!Connect(n) \lor Target!Disconnect(n))$
    $\land \text{WF}_{\langle conn, target \rangle}(Target!Start \lor Target!Stop)$
```

```
$LimitTransactions \triangleq Len(transaction) \leq NumTransactions$

$LimitTerms \triangleq$
    $\lor mastership.term < NumTerms$
    $\lor \land mastership.term = NumTerms$
      $\land mastership.master \neq Nil$

$LimitConns \triangleq$
    $\forall n \in \text{DOMAIN } conn :$
      $\lor conn[n].id < NumConns$
      $\lor \land conn[n].id = NumConns$
```

```
        ∧ conn[n].connected

TypeOK ≜
    ∧ Transaction!TypeOK
    ∧ Proposal!TypeOK
    ∧ Configuration!TypeOK
    ∧ Mastership!TypeOK

LOCAL IsOrderedChange(p, i) ≜
    ∧   history[i].type = Change
    ∧   history[i].phase = p
    ∧   ¬∃ j ∈ DOMAIN history :
            ∧ j < i
            ∧ history[j].type = Change
            ∧ history[j].phase = p
            ∧ history[j].index ≥ history[i].index

LOCAL IsOrderedRollback(p, i) ≜
    ∧   history[i].type = Rollback
    ∧   history[i].phase = p
    ∧   ¬∃ j ∈ DOMAIN history :
            ∧ j < i
            ∧ history[j].type = Change
            ∧ history[j].phase = p
            ∧ history[j].index > history[i].index
            ∧ ¬∃ k ∈ DOMAIN history :
                  ∧ k > j
                  ∧ k < i
                  ∧ history[k].type = Rollback
                  ∧ history[k].phase = p
                  ∧ history[k].index = history[j].index

Order ≜
    ∧ ∀ i ∈ DOMAIN history :
        ∨ IsOrderedChange(Commit, i)
        ∨ IsOrderedChange(Apply, i)
        ∨ IsOrderedRollback(Commit, i)
        ∨ IsOrderedRollback(Apply, i)
    ∧ ∀ i ∈ DOMAIN proposal :
          ∧ proposal[i].change.phase = Apply
          ∧ proposal[i].change.state  = Failed
          ∧ proposal[i].rollback.phase = Apply ⇒ proposal[i].rollback.state ≠ Complete
          ⇒ ∀ j ∈ DOMAIN proposal : (j > i ⇒
               (proposal[j].change.phase = Apply ⇒
                  $proposal[j].change.state \in \{Nil, Pending, Aborted\}))$

$Consistency \triangleq$
    $\land \forall i \in \textsc{domain}\ proposal :$
         $\lor configuration.committed.index < i$
         $\lor configuration.committed.revision < i$
         $\Rightarrow \neg\exists p \in \textsc{domain}\ configuration.committed.values :$
                $configuration.committed.values[p].index = i$
    $\land \forall i \in \textsc{domain}\ proposal :$
         $\lor configuration.applied.index < i$
         $\lor configuration.applied.revision < i$
         $\Rightarrow \land \neg\exists p \in \textsc{domain}\ configuration.applied.values :$
                   $configuration.applied.values[p].index = i$
             $\land \neg\exists p \in \textsc{domain}\ target.values :$
                   $target.values[p].index = i$
    $\land \land target.running$
       $\land configuration.applied.target = target.id$
       $\land configuration.state = Complete$
       $\Rightarrow \forall i \in \textsc{domain}\ proposal :$
             $\land configuration.applied.index \geq i$
             $\land configuration.applied.revision \geq i$
             $\Rightarrow \forall p \in \textsc{domain}\ proposal[i].change.values :$
                   $\land \neg\exists j \in \textsc{domain}\ proposal :$
                          $\land j > i$
                          $\land configuration.applied.index \geq j$
                          $\land configuration.applied.revision \geq j$
                   $\Rightarrow \land p \in \textsc{domain}\ target.values$
                       $\land target.values[p].value = proposal[i].change.values[p].value$
                       $\land target.values[p].index = proposal[i].change.values[p].index$

$Safety \triangleq \Box(Order \land Consistency)$

$\textsc{theorem}\ Spec \Rightarrow Safety$

$Terminates(i) \triangleq$
    $\land i \in \textsc{domain}\ transaction$
    $\land transaction[i].commit \in Done$
    $\land transaction[i].apply \in Done$
    $\land transaction[i].index \in \textsc{domain}\ proposal$
    $\land \lor \land transaction[i].type = Change$
          $\land \lor \land proposal[transaction[i].index].change.phase = Commit$
                 $\land proposal[transaction[i].index].change.state \in \{Aborted, Failed\}$
             $\lor \land proposal[transaction[i].index].change.phase = Apply$
                 $\land proposal[transaction[i].index].change.state \in Done$
       $\lor \land transaction[i].type = Rollback$
          $\land \lor \land proposal[transaction[i].index].rollback.phase = Commit$
```

$$\begin{array}{ll} & \land\ \mathit{proposal}[\mathit{transaction}[i].\mathit{index}].\mathit{rollback}.\mathit{state} \ \in \{\mathit{Aborted},\ \mathit{Failed}\} \\ \lor & \land\ \mathit{proposal}[\mathit{transaction}[i].\mathit{index}].\mathit{rollback}.\mathit{phase} = \mathit{Apply} \\ & \land\ \mathit{proposal}[\mathit{transaction}[i].\mathit{index}].\mathit{rollback}.\mathit{state} \ \in \mathit{Done} \end{array}$$

$\mathit{Termination} \triangleq$

$\quad \forall\, i \in 1 \mathrel{..} \mathit{NumTransactions} : \Diamond \mathit{Terminates}(i)$

$\mathit{Liveness} \triangleq \mathit{Termination}$

THEOREM $\mathit{Spec} \Rightarrow \mathit{Liveness}$