```
──────────────── MODULE Config ────────────────
INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

INSTANCE TLC

───────────────────────────────────────────────

GenerateTestCases ≜ FALSE

Nil ≜ "<nil>"

Change ≜ "Change"
Rollback ≜ "Rollback"

Commit ≜ "Commit"
Apply ≜ "Apply"

Pending ≜ "Pending"
Complete ≜ "Complete"
Canceled ≜ "Canceled"
Aborted ≜ "Aborted"
Failed ≜ "Failed"
Done ≜ {Complete, Canceled, Aborted, Failed}

Node ≜ {"node1"}

NumTransactions ≜ 3
NumTerms ≜ 1
NumConns ≜ 1
NumStarts ≜ 1

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*

$$\mathit{vars} \triangleq \langle \mathit{transaction}, \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{conn}, \mathit{target}, \mathit{history} \rangle$$

$$\text{LOCAL } \mathit{Transaction} \triangleq \text{INSTANCE } \mathit{Transaction}$$

$$\text{LOCAL } \mathit{Configuration} \triangleq \text{INSTANCE } \mathit{Configuration}$$

$$\text{LOCAL } \mathit{Mastership} \triangleq \text{INSTANCE } \mathit{Mastership}$$

$$\text{LOCAL } \mathit{Target} \triangleq \text{INSTANCE } \mathit{Target}$$

$$\begin{array}{l} \mathit{AppendChange}(p, v) \triangleq \\ \quad \land \mathit{Transaction}!\mathit{AppendChange}(p, v) \end{array}$$

$$\begin{array}{l} \mathit{RollbackChange}(i) \triangleq \\ \quad \land \mathit{Transaction}!\mathit{RollbackChange}(i) \end{array}$$

$$\begin{array}{l} \mathit{ReconcileTransaction}(n, i) \triangleq \\ \quad \land \mathit{Transaction}!\mathit{ReconcileTransaction}(n, i) \\ \quad \land \mathit{GenerateTestCases} \Rightarrow \mathit{Transaction}!\mathit{Test}!\mathit{Log}([\mathit{node} \mapsto n, \mathit{index} \mapsto i]) \end{array}$$

$$\begin{array}{l} \mathit{ReconcileConfiguration}(n) \triangleq \\ \quad \land \mathit{Configuration}!\mathit{ReconcileConfiguration}(n) \\ \quad \land \text{UNCHANGED } \langle \mathit{transaction}, \mathit{proposal}, \mathit{history} \rangle \\ \quad \land \mathit{GenerateTestCases} \Rightarrow \mathit{Configuration}!\mathit{Test}!\mathit{Log}([\mathit{node} \mapsto n]) \end{array}$$

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

$$\begin{array}{l} \mathit{ConnectNode}(n) \triangleq \\ \quad \land \mathit{Target}!\mathit{Connect}(n) \\ \quad \land \text{UNCHANGED } \langle \mathit{transaction}, \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle \end{array}$$

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

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

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

Formal specification, constraints, and theorems.

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

```
                    index ↦ 0,
                    value ↦ Nil]]],
            applied ↦ [
                target   ↦ 0,
                index    ↦ 0,
                revision ↦ 0,
                values   ↦ [
                    p ∈ {}  ↦ [
                        index ↦ 0,
                        value ↦ Nil]]]]
    ∧ target = [
            id       ↦ 1,
            running  ↦ TRUE,
            values   ↦ [
                p ∈ {}  ↦ [
                    index ↦ 0,
                    value ↦ Nil]]]
    ∧ mastership = [
            master ↦ CHOOSE n ∈ Node : TRUE,
            term   ↦ 1,
            conn   ↦ 1]
    ∧ conn = [
            n ∈ Node ↦ [
                id        ↦ 1,
                connected ↦ TRUE]]
    ∧ history = ⟨⟩

Next ≜
    ∨ ∃ p ∈ Path, v ∈ Value :
          AppendChange(p, v)
    ∨ ∃ i ∈ 1 .. NumTransactions :
          RollbackChange(i)
    ∨ ∃ n ∈ Node :
          ∃ i ∈ 1 .. NumTransactions :
            ReconcileTransaction(n, i)
    ∨ ∃ n ∈ Node :
          ReconcileConfiguration(n)
    ∨ ∃ n ∈ Node :
          ReconcileMastership(n)
    ∨ ∃ n ∈ Node :
          ∨ ConnectNode(n)
          ∨ DisconnectNode(n)
    ∨ StartTarget
    ∨ StopTarget
```

```
Spec ≜
    ∧ Init
    ∧ □[Next]_vars
    ∧ ∀ i ∈ 1 . . NumTransactions :
         WF_⟨transaction⟩(Transaction!RollbackChange(i))
    ∧ ∀ n ∈ Node, i ∈ 1 . . NumTransactions :
         WF_⟨transaction, proposal, configuration, mastership, conn, target, history⟩(Transaction!ReconcileTransaction(n, i))
    ∧ ∀ n ∈ Node :
         WF_⟨configuration, mastership, conn, target⟩(Configuration!ReconcileConfiguration(n))

Alias ≜ [
   log ↦ [
      i ∈ DOMAIN transaction ↦ [
         change ↦
            IF transaction[i].change.proposal ≠ 0 THEN
               [commit ↦ proposal[transaction[i].change.proposal].commit,
                apply  ↦ proposal[transaction[i].change.proposal].apply,
                values ↦ transaction[i].change.values]
             ELSE
               [commit ↦ Nil,
                apply  ↦ Nil,
                values ↦ transaction[i].change.values],
         rollback ↦
            IF transaction[i].rollback.proposal ≠ 0 THEN
               [commit ↦ proposal[transaction[i].rollback.proposal].commit,
                apply  ↦ proposal[transaction[i].rollback.proposal].apply,
                values ↦ transaction[i].rollback.values]
             ELSE
               [commit ↦ Nil,
                apply  ↦ Nil,
                values ↦ transaction[i].rollback.values]] @@
         transaction[i]],
   transaction   ↦ transaction,
   proposal      ↦ proposal,
   configuration ↦ configuration,
   mastership     ↦ mastership,
   conn          ↦ conn,
   target        ↦ target,
   history       ↦ history]
```

```
LimitTransactions ≜ Len(transaction) ≤ NumTransactions

LimitTerms ≜
    ∨ mastership.term < NumTerms
```

```
   ∨ ∧ mastership.term = NumTerms
     ∧ mastership.master ≠ Nil

LimitConns ≜
  ∀ n ∈ DOMAIN conn :
     ∨ conn[n].id < NumConns
     ∨ ∧ conn[n].id = NumConns
       ∧ conn[n].connected

LimitStarts ≜
   ∨ target.id < 2
   ∨ ∧ target.id = 2
     ∧ target.running

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

TypeOK ≜
   ∧ Transaction!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].index = history[i].index
   ∧   ¬∃ 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 transaction :
        ∧ transaction[i].change.proposal ≠ 0
        ∧ proposal[transaction[i].change.proposal].apply = Failed
        ∧ transaction[i].rollback.proposal ≠ 0 ⇒
              proposal[transaction[i].rollback.proposal].apply ≠ Complete
        ⇒ ∀ j ∈ DOMAIN transaction : (j > i ⇒
             (transaction[j].change.proposal ≠ 0 ⇒
                proposal[transaction[j].change.proposal].apply ≠ Complete))

LOCAL IsChangeCommitted(i) ≜
  ∧   transaction[i].change.proposal ≠ 0
  ∧   proposal[transaction[i].change.proposal].commit = Complete
  ∧   transaction[i].rollback.proposal ≠ 0 ⇒
          proposal[transaction[i].rollback.proposal].commit ≠ Complete

LOCAL IsChangeApplied(i) ≜
  ∧   transaction[i].change.proposal ≠ 0
  ∧   proposal[transaction[i].change.proposal].apply = Complete
  ∧   transaction[i].rollback.proposal ≠ 0 ⇒
          proposal[transaction[i].rollback.proposal].apply ≠ Complete

Consistency ≜
  ∧ ∀ i ∈ DOMAIN transaction :
       ∧ IsChangeCommitted(i)
       ∧ ¬∃ j ∈ DOMAIN transaction :
              ∧ j > i
              ∧ IsChangeCommitted(j)
       ⇒ ∀ p ∈ DOMAIN transaction[i].change.values :
             ∧ configuration.committed.values[p] = transaction[i].change.values[p]
  ∧ ∀ i ∈ DOMAIN transaction :
       ∧ IsChangeApplied(i)
       ∧ ¬∃ j ∈ DOMAIN transaction :
              ∧ j > i
              ∧ IsChangeApplied(j)
       ⇒ ∀ p ∈ DOMAIN transaction[i].change.values :
             ∧ configuration.applied.values[p] = transaction[i].change.values[p]
             ∧ ∧ target.running
               ∧ configuration.applied.target = target.id
```

```
                 $\land\ configuration.state = Complete$
                 $\Rightarrow target.values[p] = transaction[i].change.values[p]$

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

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

$\textsc{local}\ IsChanging(i) \triangleq$
    $\land\quad i \in \textsc{domain}\ transaction$
    $\land\quad transaction[i].phase = Change$

$\textsc{local}\ IsChanged(i) \triangleq$
    $\land\quad i \in \textsc{domain}\ transaction$
    $\land\quad transaction[i].change.proposal \in \textsc{domain}\ proposal$
    $\land\quad proposal[transaction[i].change.proposal].commit \in Done$
    $\land\quad proposal[transaction[i].change.proposal].apply \in Done$

$\textsc{local}\ IsRollingBack(i) \triangleq$
    $\land\quad i \in \textsc{domain}\ transaction$
    $\land\quad transaction[i].phase = Rollback$

$\textsc{local}\ IsRolledBack(i) \triangleq$
    $\land\quad i \in \textsc{domain}\ transaction$
    $\land\quad transaction[i].rollback.proposal \in \textsc{domain}\ proposal$
    $\land\quad proposal[transaction[i].rollback.proposal].commit \in Done$
    $\land\quad proposal[transaction[i].rollback.proposal].apply \in Done$

$Terminates(i) \triangleq$
    $\land\ IsChanging(i) \leadsto IsChanged(i)$
    $\land\ IsRollingBack(i) \leadsto IsRolledBack(i)$

$Termination \triangleq$
    $\forall\, i \in 1 \mathrel{..} NumTransactions : Terminates(i)$

$Liveness \triangleq Termination$

$\textsc{theorem}\ Spec \Rightarrow Liveness$
```