```
┌──────────────────────── 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"
Canceled ≜ "Canceled"
Failed ≜ "Failed"

Node ≜ {"node1"}

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

Path ≜ {"path1"}
Value ≜ {"value1", "value2"}

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

  A transaction log.
VARIABLE transactions

  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 ≜ ⟨transactions, configuration, mastership, conn, target, history⟩

LOCAL Transaction ≜ INSTANCE Transaction

LOCAL Configuration ≜ INSTANCE Configuration

LOCAL Mastership ≜ INSTANCE Mastership

LOCAL Target ≜ INSTANCE Target

AppendChange(i) ≜
   ∧ Transaction!AppendChange(i)

RollbackChange(i) ≜
   ∧ Transaction!RollbackChange(i)

ReconcileTransaction(n, i) ≜
   ∧ i ∈ DOMAIN transactions
   ∧ ∨ ∧ Transaction!ReconcileTransaction(n, i)
       ∧ GenerateTestCases ⇒ Transaction!Test!Log([node ↦ n, index ↦ i])
     ∨ ∧ GenerateTestCases
       ∧ ¬ENABLED Transaction!ReconcileTransaction(n, i)
       ∧ UNCHANGED vars
       ∧ Transaction!Test!Log([node ↦ n, index ↦ i])

ReconcileConfiguration(n) ≜
   ∨ ∧ Configuration!ReconcileConfiguration(n)
     ∧ UNCHANGED ⟨transactions, history⟩
     ∧ GenerateTestCases ⇒ Configuration!Test!Log([node ↦ n])
   ∨ ∧ GenerateTestCases
     ∧ ¬ENABLED Configuration!ReconcileConfiguration(n)
     ∧ UNCHANGED vars
     ∧ Configuration!Test!Log([node ↦ n])

ReconcileMastership(n) ≜
   ∨ ∧ Mastership!ReconcileMastership(n)
     ∧ UNCHANGED ⟨transactions, configuration, target, history⟩
```

```
        ∧ GenerateTestCases ⇒ Mastership!Test!Log([node ↦ n])
    ∨ ∧ GenerateTestCases
      ∧ ¬ENABLED Mastership!ReconcileMastership(n)
      ∧ UNCHANGED vars
      ∧ Mastership!Test!Log([node ↦ n])

ConnectNode(n) ≜
    ∧ Target!Connect(n)
    ∧ UNCHANGED ⟨transactions, configuration, mastership, history⟩

DisconnectNode(n) ≜
    ∧ Target!Disconnect(n)
    ∧ UNCHANGED ⟨transactions, configuration, mastership, history⟩

StartTarget ≜
    ∧ Target!Start
    ∧ UNCHANGED ⟨transactions, configuration, mastership, history⟩

StopTarget ≜
    ∧ Target!Stop
    ∧ UNCHANGED ⟨transactions, configuration, mastership, history⟩
```

Formal specification, constraints, and theorems.

```
Init ≜
    ∧ transactions = [
          i ∈ {} ↦ [
            phase    ↦ Nil,
            values ↦ [
                p ∈ {} ↦ Nil],
            change   ↦ [
                commit ↦ Nil,
                apply  ↦ Nil],
            rollback ↦ [
                commit ↦ Nil,
                apply  ↦ Nil]]]
    ∧ configuration = [
          state  ↦ Pending,
          term   ↦ 0,
          committed ↦ [
              index    ↦ 0,
              change   ↦ 0,
              target   ↦ 0,
              ordinal  ↦ 0,
              revision ↦ 0,
              values   ↦ [
                 p ∈ {} ↦ Nil]],
          applied ↦ [
              index    ↦ 0,
              target   ↦ 0,
              ordinal  ↦ 0,
              revision ↦ 0,
              values   ↦ [
                 p ∈ {} ↦ 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 ≜
    ∨ ∃ i ∈ 1 . . NumTransactions :
         ∨ AppendChange(i)
         ∨ 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 :
```

$$
\begin{array}{l}
\quad\quad \mathrm{WF}_{\langle \mathit{transactions} \rangle}(\mathit{Transaction}\,!\,\mathit{RollbackChange}(i))\\
\quad \land \forall n \in \mathit{Node},\ i \in 1 \mathrel{..} \mathit{NumTransactions} :\\
\quad\quad \mathrm{WF}_{\langle \mathit{transactions},\ \mathit{configuration},\ \mathit{mastership},\ \mathit{conn},\ \mathit{target},\ \mathit{history} \rangle}(\mathit{Transaction}\,!\,\mathit{ReconcileTransaction}(n,\ i))\\
\quad \land \forall n \in \mathit{Node} :\\
\quad\quad \mathrm{WF}_{\langle \mathit{configuration},\ \mathit{mastership},\ \mathit{conn},\ \mathit{target} \rangle}(\mathit{Configuration}\,!\,\mathit{ReconcileConfiguration}(n))\\
\quad \land \forall n \in \mathit{Node} :\\
\quad\quad \mathrm{WF}_{\langle \mathit{mastership},\ \mathit{conn} \rangle}(\mathit{Mastership}\,!\,\mathit{ReconcileMastership}(n))\\
\quad \land \forall n \in \mathit{Node} :\\
\quad\quad \mathrm{WF}_{\langle \mathit{conn},\ \mathit{target} \rangle}(\mathit{Target}\,!\,\mathit{Connect}(n) \lor \mathit{Target}\,!\,\mathit{Disconnect}(n))\\
\quad \land \mathrm{WF}_{\langle \mathit{conn},\ \mathit{target} \rangle}(\mathit{Target}\,!\,\mathit{Start} \lor \mathit{Target}\,!\,\mathit{Stop})
\end{array}
$$

$$
\begin{array}{l}
\mathit{LimitTerms} \triangleq\\
\quad \lor \mathit{mastership.term} < \mathit{NumTerms}\\
\quad \lor \land \mathit{mastership.term} = \mathit{NumTerms}\\
\quad\quad \land \mathit{mastership.master} \neq \mathit{Nil}
\end{array}
$$

$$
\begin{array}{l}
\mathit{LimitConns} \triangleq\\
\quad \forall n \in \textsc{domain}\ \mathit{conn} :\\
\quad\quad \lor \mathit{conn}[n].\mathit{id} < \mathit{NumConns}\\
\quad\quad \lor \land \mathit{conn}[n].\mathit{id} = \mathit{NumConns}\\
\quad\quad\quad \land \mathit{conn}[n].\mathit{connected}
\end{array}
$$

$$
\begin{array}{l}
\mathit{LimitStarts} \triangleq\\
\quad \lor \mathit{target.id} < 2\\
\quad \lor \land \mathit{target.id} = 2\\
\quad\quad \land \mathit{target.running}
\end{array}
$$

$$
\begin{array}{l}
\mathit{TypeOK} \triangleq\\
\quad \land \mathit{Transaction}\,!\,\mathit{TypeOK}\\
\quad \land \mathit{Configuration}\,!\,\mathit{TypeOK}\\
\quad \land \mathit{Mastership}\,!\,\mathit{TypeOK}
\end{array}
$$

$$
\begin{array}{l}
\mathit{StatusCommitted}(i) \triangleq\\
\quad \lor \land \mathit{transactions}'[i].\mathit{change.commit} \notin \{\mathit{Pending},\ \mathit{Canceled}\}\\
\quad\quad \land \mathit{transactions}[i].\mathit{change.commit} \neq \mathit{transactions}'[i].\mathit{change.commit}\\
\quad \lor \land \mathit{transactions}'[i].\mathit{rollback.commit} \notin \{\mathit{Pending},\ \mathit{Canceled}\}\\
\quad\quad \land \mathit{transactions}[i].\mathit{rollback.commit} \neq \mathit{transactions}'[i].\mathit{rollback.commit}
\end{array}
$$

$$
\begin{array}{l}
\mathit{StatusApplied}(i) \triangleq\\
\quad \lor \land \mathit{transactions}'[i].\mathit{change.apply} \notin \{\mathit{Pending},\ \mathit{Canceled}\}\\
\quad\quad \land \mathit{transactions}[i].\mathit{change.apply} \neq \mathit{transactions}'[i].\mathit{change.apply}\\
\quad \lor \land \mathit{transactions}'[i].\mathit{rollback.apply} \notin \{\mathit{Pending},\ \mathit{Canceled}\}\\
\quad\quad \land \mathit{transactions}[i].\mathit{rollback.apply} \neq \mathit{transactions}'[i].\mathit{rollback.apply}
\end{array}
$$

```
ValidStatus(t, i, j) ≜
   ∧ j ∈ DOMAIN history
   ∧ history[j].index = i
   ∧ ∨ ∧ history[j].type = Change
         ∧ history[j].phase = Commit
         ∧ t[i].change.commit = history[j].status
      ∨ ∧ history[j].type = Change
         ∧ history[j].phase = Apply
         ∧ t[i].change.apply = history[j].status
      ∨ ∧ history[j].type = Rollback
         ∧ history[j].phase = Commit
         ∧ t[i].rollback.commit = history[j].status
      ∨ ∧ history[j].type = Rollback
         ∧ history[j].phase = Apply
         ∧ t[i].rollback.apply = history[j].status

ValidCommit(t, i) ≜
   LET j ≜ CHOOSE j ∈ DOMAIN history :
                  ∧ history[j].phase = Commit
                  ∧ ¬∃ k ∈ DOMAIN history :
                        ∧ history[k].phase = Commit
                        ∧ k > j
   IN  ValidStatus(t, i, j)

ValidApply(t, i) ≜
   LET j ≜ CHOOSE j ∈ DOMAIN history :
                  ∧ history[j].phase = Apply
                  ∧ ¬∃ k ∈ DOMAIN history :
                        ∧ history[k].phase = Apply
                        ∧ k > j
   IN  ValidStatus(t, i, j)

ConfigurationCommitted ≜
   ∧ configuration′.committed ≠ configuration.committed
   ∧ ∃ i ∈ DOMAIN history : history[i].phase = Commit
   ⇒ LET i ≜ CHOOSE i ∈ DOMAIN history :
                     ∧ history[i].phase = Commit
                     ∧ ¬∃ j ∈ DOMAIN history :
                           ∧ history[j].phase = Commit
                           ∧ j > i
      IN  ValidStatus(transactions, history[i].index, i)

ConfigurationApplied ≜
   ∧ configuration′.applied ≠ configuration.applied
   ∧ ∃ i ∈ DOMAIN history : history[i].phase = Apply
   ⇒ LET i ≜ CHOOSE i ∈ DOMAIN history :
```

```
                    $\land\ history[i].phase = Apply$
                    $\land\ \neg\exists\, j \in \text{DOMAIN}\ history :$
                          $\land\ history[j].phase = Apply$
                          $\land\ j > i$
       $\text{IN}\quad ValidStatus(transactions,\ history[i].index,\ i)$

$StatusChanged\ \triangleq$
  $\forall\, i \in 1 \mathrel{..} NumTransactions :$
    $\land\ i \in \text{DOMAIN}\ transactions \Rightarrow$
        $\land\ StatusCommitted(i) \Rightarrow ValidCommit(transactions',\ i)$
        $\land\ StatusApplied(i) \Rightarrow ValidApply(transactions',\ i)$

$Transition\ \triangleq\ \Box[ConfigurationCommitted \land ConfigurationApplied \land StatusChanged]_{\langle transactions,\, history\rangle}$

$\text{LOCAL}\ IsOrderedChange(p,\ i)\ \triangleq$
   $\land\quad history[i].type = Change$
   $\land\quad history[i].phase = p$
   $\land\quad history[i].status = Complete$
   $\land\quad \neg\exists\, j \in \text{DOMAIN}\ history :$
               $\land\ j < i$
               $\land\ history[j].type = Change$
               $\land\ history[j].phase = p$
               $\land\ history[j].status = Complete$
               $\land\ history[j].index \geq history[i].index$

$\text{LOCAL}\ IsOrderedRollback(p,\ i)\ \triangleq$
   $\land\quad history[i].type = Rollback$
   $\land\quad history[i].phase = p$
   $\land\quad history[i].status = Complete$
   $\land\quad \exists\, j \in \text{DOMAIN}\ history :$
             $\land\ j < i$
             $\land\ history[j].type = Change$
             $\land\ history[j].status = Complete$
             $\land\ history[j].index = history[i].index$
   $\land\quad \neg\exists\, j \in \text{DOMAIN}\ history :$
               $\land\ j < i$
               $\land\ history[j].type = Change$
               $\land\ history[j].phase = p$
               $\land\ history[j].status = Complete$
               $\land\ history[j].index > history[i].index$
               $\land\ \neg\exists\, k \in \text{DOMAIN}\ history :$
                     $\land\ k > j$
                     $\land\ k < i$
                     $\land\ history[k].type = Rollback$
                     $\land\ history[k].phase = p$
                     $\land\ history[j].status = Complete$
```

```
                $\land\ history[k].index = history[j].index$

$Order \triangleq$
    $\land\ \forall\, i \in$ DOMAIN $history$ :
        $history[i].status = Complete \Rightarrow$
            $\lor\ IsOrderedChange(Commit,\ i)$
            $\lor\ IsOrderedChange(Apply,\ i)$
            $\lor\ IsOrderedRollback(Commit,\ i)$
            $\lor\ IsOrderedRollback(Apply,\ i)$
    $\land\ \forall\, i \in$ DOMAIN $transactions$ :
        $\land\ transactions[i].change.apply = Failed$
        $\land\ transactions[i].rollback.apply \neq Complete$
        $\Rightarrow \neg\exists\, j \in$ DOMAIN $transactions$ :
                $\land\ j > i$
                $\land\ transactions[i].change.apply \in \{InProgress,\ Complete\}$

LOCAL $IsChangeCommitted(i) \triangleq$
    $\land\ configuration.committed.revision = i$

LOCAL $IsChangeApplied(i) \triangleq$
    $\land\ configuration.applied.revision = i$

$Consistency \triangleq$
    $\land\ \forall\, i \in$ DOMAIN $transactions$ :
        $\land\ IsChangeCommitted(i)$
        $\land\ \neg\exists\, j \in$ DOMAIN $transactions$ :
                $\land\ j > i$
                $\land\ IsChangeCommitted(j)$
        $\Rightarrow \forall\, p \in$ DOMAIN $transactions[i].change.values$ :
            $\land\ configuration.committed.values[p] = transactions[i].change.values[p]$
    $\land\ \forall\, i \in$ DOMAIN $transactions$ :
        $\land\ IsChangeApplied(i)$
        $\land\ \neg\exists\, j \in$ DOMAIN $transactions$ :
                $\land\ j > i$
                $\land\ IsChangeApplied(j)$
        $\Rightarrow \forall\, p \in$ DOMAIN $transactions[i].change.values$ :
            $\land\ configuration.applied.values[p] = transactions[i].change.values[p]$
            $\land\ \land\ target.running$
                $\land\ configuration.applied.target = target.id$
                $\land\ configuration.state = Complete$
                $\Rightarrow target.values[p] = transactions[i].change.values[p]$

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

THEOREM $Spec \Rightarrow Safety$

LOCAL $IsChanging(i) \triangleq$
```

$\quad \land \quad i \in \textsc{domain}\ transactions$
$\quad \land \quad transactions[i].phase = Change$

$\textsc{local}\ IsChanged(i) \triangleq$
$\quad \land \quad i \in \textsc{domain}\ transactions$
$\quad \land \quad transactions[i].change.commit \in \{Complete, Failed\}$
$\quad \land \quad transactions[i].change.apply \in \{Complete, Aborted, Failed\}$

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

$\textsc{local}\ IsRolledBack(i) \triangleq$
$\quad \land \quad i \in \textsc{domain}\ transactions$
$\quad \land \quad transactions[i].rollback.commit \in \{Complete, Failed\}$
$\quad \land \quad transactions[i].rollback.apply \in \{Complete, Aborted, Failed\}$

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

$Termination \triangleq$
$\quad \forall\, i \in 1 \,..\, NumTransactions : Terminates(i)$

$Liveness \triangleq Termination$

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