MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

GenerateTestCases $\triangleq$ FALSE

Nil $\triangleq$ "<nil>"

Change $\triangleq$ "Change"
Rollback $\triangleq$ "Rollback"

Commit $\triangleq$ "Commit"
Apply $\triangleq$ "Apply"

Pending $\triangleq$ "Pending"
InProgress $\triangleq$ "InProgress"
Complete $\triangleq$ "Complete"
Aborted $\triangleq$ "Aborted"
Canceled $\triangleq$ "Canceled"
Failed $\triangleq$ "Failed"

Node $\triangleq$ {"node1"}

NumTransactions $\triangleq$ 3
NumTerms $\triangleq$ 1
NumConns $\triangleq$ 1
NumStarts $\triangleq$ 1

Path $\triangleq$ {"path1"}
Value $\triangleq$ {"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) ≜
   ∨ ∧ 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,
           transaction ↦ 0,
           revision      ↦ 0,
           values        ↦ [
             p ∈ {}    ↦ Nil]],
        applied ↦ [
           index       ↦ 0,
           target      ↦ 0,
           ordinal     ↦ 0,
           transaction ↦ 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 ∈ DOMAIN transactions :
        ReconcileTransaction(n, i)
  ∨ ∃ n ∈ Node :
        ReconcileConfiguration(n)
  ∨ ∃ n ∈ Node :
        ReconcileMastership(n)
  ∨ ∃ n ∈ Node :
        ∨ ConnectNode(n)
        ∨ DisconnectNode(n)
  ∨ StartTarget
  ∨ StopTarget

Spec ≜
  ∧ Init
  ∧ □[Next]_vars
```

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

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

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

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

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

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

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

```
        $\land \mathit{transactions}[i].\mathit{rollback}.\mathit{apply} \neq \mathit{transactions}'[i].\mathit{rollback}.\mathit{apply}$

$\mathit{ValidStatus}(t, i, j) \triangleq$
    $\land j \in \textsc{domain}\ \mathit{history}$
    $\land \mathit{history}[j].\mathit{index} = i$
    $\land \lor \land \mathit{history}[j].\mathit{type} = \mathit{Change}$
            $\land \mathit{history}[j].\mathit{phase} = \mathit{Commit}$
            $\land t[i].\mathit{change}.\mathit{commit} = \mathit{history}[j].\mathit{status}$
       $\lor \land \mathit{history}[j].\mathit{type} = \mathit{Change}$
            $\land \mathit{history}[j].\mathit{phase} = \mathit{Apply}$
            $\land t[i].\mathit{change}.\mathit{apply} = \mathit{history}[j].\mathit{status}$
       $\lor \land \mathit{history}[j].\mathit{type} = \mathit{Rollback}$
            $\land \mathit{history}[j].\mathit{phase} = \mathit{Commit}$
            $\land t[i].\mathit{rollback}.\mathit{commit} = \mathit{history}[j].\mathit{status}$
       $\lor \land \mathit{history}[j].\mathit{type} = \mathit{Rollback}$
            $\land \mathit{history}[j].\mathit{phase} = \mathit{Apply}$
            $\land t[i].\mathit{rollback}.\mathit{apply} = \mathit{history}[j].\mathit{status}$

$\mathit{ValidCommit}(t, i) \triangleq$
    $\textsc{let}\ j \triangleq \textsc{choose}\ j \in \textsc{domain}\ \mathit{history} :$
                    $\land \mathit{history}[j].\mathit{phase} = \mathit{Commit}$
                    $\land \neg\exists\, k \in \textsc{domain}\ \mathit{history} :$
                           $\land \mathit{history}[k].\mathit{phase} = \mathit{Commit}$
                           $\land k > j$
    $\textsc{in}\quad \mathit{ValidStatus}(t, i, j)$

$\mathit{ValidApply}(t, i) \triangleq$
    $\textsc{let}\ j \triangleq \textsc{choose}\ j \in \textsc{domain}\ \mathit{history} :$
                    $\land \mathit{history}[j].\mathit{phase} = \mathit{Apply}$
                    $\land \neg\exists\, k \in \textsc{domain}\ \mathit{history} :$
                           $\land \mathit{history}[k].\mathit{phase} = \mathit{Apply}$
                           $\land k > j$
    $\textsc{in}\quad \mathit{ValidStatus}(t, i, j)$

$\mathit{ConfigurationCommitted} \triangleq$
    $\land \mathit{configuration}'.\mathit{committed} \neq \mathit{configuration}.\mathit{committed}$
    $\land \exists\, i \in \textsc{domain}\ \mathit{history} : \mathit{history}[i].\mathit{phase} = \mathit{Commit}$
    $\Rightarrow \textsc{let}\ i \triangleq \textsc{choose}\ i \in \textsc{domain}\ \mathit{history} :$
                        $\land \mathit{history}[i].\mathit{phase} = \mathit{Commit}$
                        $\land \neg\exists\, j \in \textsc{domain}\ \mathit{history} :$
                               $\land \mathit{history}[j].\mathit{phase} = \mathit{Commit}$
                               $\land j > i$
        $\textsc{in}\quad \mathit{ValidStatus}(\mathit{transactions}, \mathit{history}[i].\mathit{index}, i)$

$\mathit{ConfigurationApplied} \triangleq$
    $\land \mathit{configuration}'.\mathit{applied} \neq \mathit{configuration}.\mathit{applied}$
```

```
  $\land \exists i \in \textsc{domain}\ history : history[i].phase = Apply$
  $\Rightarrow \textsc{let}\ i \triangleq \textsc{choose}\ i \in \textsc{domain}\ history :$
                    $\land history[i].phase = Apply$
                    $\land \neg\exists j \in \textsc{domain}\ history :$
                          $\land history[j].phase = Apply$
                          $\land j > i$
     $\textsc{in}\quad ValidStatus(transactions,\ history[i].index,\ i)$
```

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

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

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

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

```
                    $\land\ history[k].phase = p$
                    $\land\ history[j].status = Complete$
                    $\land\ history[k].index = history[j].index$

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

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

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

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

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

THEOREM $Spec \Rightarrow Safety$
```

```
LOCAL IsChanging(i) ≜
    ∧  i ∈ DOMAIN transactions
    ∧  transactions[i].phase = Change

LOCAL IsChanged(i) ≜
    ∧  i ∈ DOMAIN transactions
    ∧  transactions[i].change.commit ∈ {Complete, Failed}
    ∧  transactions[i].change.apply ∈ {Complete, Aborted, Failed}

LOCAL IsRollingBack(i) ≜
    ∧  i ∈ DOMAIN transactions
    ∧  transactions[i].phase = Rollback

LOCAL IsRolledBack(i) ≜
    ∧  i ∈ DOMAIN transactions
    ∧  transactions[i].rollback.commit ∈ {Complete, Failed}
    ∧  transactions[i].rollback.apply ∈ {Complete, Aborted, Failed}

Terminates(i) ≜
    ∧ IsChanging(i) ↝ IsChanged(i)
    ∧ IsRollingBack(i) ↝ IsRolledBack(i)

Termination ≜
    ∀ i ∈ 1 . . NumTransactions : Terminates(i)

Liveness ≜ Termination

THEOREM Spec ⇒ Liveness
```