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) ≜
   ∧ Transaction!ReconcileTransaction(n, i)
   ∧ GenerateTestCases ⇒ Transaction!Test!Log([node ↦ n, index ↦ i])

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

ReconcileMastership(n) ≜
   ∧ Mastership!ReconcileMastership(n)
   ∧ UNCHANGED ⟨transactions, configuration, target, history⟩
   ∧ GenerateTestCases ⇒ 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,
             maxIndex    ↦ 0,
             target      ↦ 0,
             seqnum      ↦ 0,
             transaction ↦ 0,
             revision    ↦ 0,
             values      ↦ [
                p ∈ {}   ↦ Nil]],
          applied ↦ [
             index       ↦ 0,
             target      ↦ 0,
             seqnum      ↦ 0,
             transaction ↦ 0,
             revision    ↦ 0,
             values      ↦ [
                p ∈ {}   ↦ Nil]]]
    ∧ target = [
          id      ↦ 1,
          running ↦ TRUE,
        $values \mapsto [$
            $p \in \{\} \mapsto [$
                $index \mapsto 0,$
                $value \mapsto Nil]]]$
    $\land\ mastership = [$
        $master \mapsto \text{CHOOSE } n \in Node : \text{TRUE},$
        $term \mapsto 1,$
        $conn \mapsto 1]$
    $\land\ conn = [$
        $n \in Node \mapsto [$
            $id \mapsto 1,$
            $connected \mapsto \text{TRUE}]]$
    $\land\ history = \langle\rangle$

$Next \triangleq$
    $\lor\ \exists\, i \in 1 \,..\, NumTransactions :$
        $\lor\ AppendChange(i)$
        $\lor\ RollbackChange(i)$
    $\lor\ \exists\, n \in Node,\ i \in \text{DOMAIN } transactions :$
        $ReconcileTransaction(n,\ i)$
    $\lor\ \exists\, n \in Node :$
        $ReconcileConfiguration(n)$
    $\lor\ \exists\, n \in Node :$
        $ReconcileMastership(n)$
    $\lor\ \exists\, n \in Node :$
        $\lor\ ConnectNode(n)$
        $\lor\ DisconnectNode(n)$
    $\lor\ StartTarget$
    $\lor\ StopTarget$

$Spec \triangleq$
    $\land\ Init$
    $\land\ \Box[Next]_{vars}$
    $\land\ \forall\, i \in 1 \,..\, NumTransactions :$
        $\text{WF}_{\langle transactions \rangle}(Transaction!RollbackChange(i))$
    $\land\ \forall\, n \in Node,\ i \in 1 \,..\, NumTransactions :$
        $\text{WF}_{\langle transactions,\ configuration,\ mastership,\ conn,\ target,\ history \rangle}(Transaction!ReconcileTransaction(n,\ i))$
    $\land\ \forall\, n \in Node :$
        $\text{WF}_{\langle configuration,\ mastership,\ conn,\ target \rangle}(Configuration!ReconcileConfiguration(n))$
    $\land\ \forall\, n \in Node :$
        $\text{WF}_{\langle mastership,\ conn \rangle}(Mastership!ReconcileMastership(n))$
    $\land\ \forall\, n \in Node :$
        $\text{WF}_{\langle conn,\ target \rangle}(Target!Connect(n) \lor Target!Disconnect(n))$
    $\land\ \text{WF}_{\langle conn,\ target \rangle}(Target!Start \lor Target!Stop)$
```

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

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

$LimitStarts \triangleq$
    $\lor target.id < 2$
    $\lor \land target.id = 2$
       $\land target.running$
```

```
$TypeOK \triangleq$
    $\land Transaction!TypeOK$
    $\land Configuration!TypeOK$
    $\land Mastership!TypeOK$

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

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

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

```
        $\lor \land history[j].type = Rollback$
          $\land history[j].phase = Apply$
          $\land t[i].rollback.apply = history[j].status$

$ValidCommit(t, i) \triangleq$
  $\text{LET } j \triangleq \text{ CHOOSE } j \in \text{ DOMAIN } history :$
                $\land history[j].phase = Commit$
                $\land \neg\exists\, k \in \text{ DOMAIN } history :$
                      $\land history[k].phase = Commit$
                      $\land k > j$
  $\text{IN} \quad ValidStatus(t, i, j)$

$ValidApply(t, i) \triangleq$
  $\text{LET } j \triangleq \text{ CHOOSE } j \in \text{ DOMAIN } history :$
                $\land history[j].phase = Apply$
                $\land \neg\exists\, k \in \text{ DOMAIN } history :$
                      $\land history[k].phase = Apply$
                      $\land k > j$
  $\text{IN} \quad ValidStatus(t, i, j)$

$ConfigurationCommitted \triangleq$
  $\land configuration'.committed \neq configuration.committed$
  $\land \exists\, i \in \text{ DOMAIN } history : history[i].phase = Commit$
  $\Rightarrow \text{LET } i \triangleq \text{ CHOOSE } i \in \text{ DOMAIN } history :$
                  $\land history[i].phase = Commit$
                  $\land \neg\exists\, j \in \text{ DOMAIN } history :$
                        $\land history[j].phase = Commit$
                        $\land j > i$
     $\text{IN} \quad ValidStatus(transactions, history[i].index, i)$

$ConfigurationApplied \triangleq$
  $\land configuration'.applied \neq configuration.applied$
  $\land \exists\, i \in \text{ DOMAIN } history : history[i].phase = Apply$
  $\Rightarrow \text{LET } i \triangleq \text{ CHOOSE } i \in \text{ DOMAIN } history :$
                  $\land history[i].phase = Apply$
                  $\land \neg\exists\, j \in \text{ DOMAIN } history :$
                        $\land history[j].phase = Apply$
                        $\land j > i$
     $\text{IN} \quad ValidStatus(transactions, history[i].index, i)$

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

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

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

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

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

```
                ∧ j > i
                ∧ transactions[i].change.apply ∈ {InProgress, Complete}

LOCAL IsChangeCommitted(i) ≜
    ∧   configuration.committed.revision = i

LOCAL IsChangeApplied(i) ≜
    ∧   configuration.applied.revision = i

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

Safety ≜ □(Order ∧ Consistency)

THEOREM Spec ⇒ 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
```

$\land \quad transactions[i].rollback.commit \in \{Complete, Failed\}$
$\land \quad transactions[i].rollback.apply \in \{Complete, Aborted, Failed\}$

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

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

$Liveness \triangleq Termination$

THEOREM $Spec \Rightarrow Liveness$