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"
Complete $\triangleq$ "Complete"
Aborted $\triangleq$ "Aborted"
Failed $\triangleq$ "Failed"
Done $\triangleq$ {*Complete*, *Aborted*, *Failed*}

Node $\triangleq$ {"node1"}

NumTransactions $\triangleq$ 3
NumTerms $\triangleq$ 2
NumConns $\triangleq$ 2
NumStarts $\triangleq$ 2

Path $\triangleq$ {"path1"}
Value $\triangleq$ {"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 $history$

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

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

LOCAL $Transaction \triangleq$ INSTANCE $Transaction$

LOCAL $Configuration \triangleq$ INSTANCE $Configuration$

LOCAL $Mastership \triangleq$ INSTANCE $Mastership$

LOCAL $Target \triangleq$ INSTANCE $Target$

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

$AppendChange(p, v) \triangleq$
    $\land Transaction!AppendChange(p, v)$

$RollbackChange(i) \triangleq$
    $\land Transaction!RollbackChange(i)$

$ReconcileTransaction(n, i) \triangleq$
    $\land i \in$ DOMAIN $transaction$
    $\land Transaction!ReconcileTransaction(n, i)$
    $\land GenerateTestCases \Rightarrow$
          LET $context \triangleq [node \mapsto n, index \mapsto i]$
          IN  $Transaction!Test!Log(context)$

$ReconcileConfiguration(n) \triangleq$
    $\land Configuration!ReconcileConfiguration(n)$
    $\land$ UNCHANGED $\langle transaction, proposal, history \rangle$
    $\land GenerateTestCases \Rightarrow Configuration!Test!Log([node \mapsto n])$

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

$ConnectNode(n) \triangleq$
    $\land Target!Connect(n)$
    ∧ UNCHANGED ⟨transaction, proposal, configuration, mastership, history⟩

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

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

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

Formal specification, constraints, and theorems.

```
Init ≜
    ∧ transaction = [
          i ∈ {} ↦ [
            phase  ↦ Nil,
            change ↦ [
                proposal ↦ 0,
                revision ↦ 0,
                values   ↦ [
                    p ∈ {}  ↦ [
                       index ↦ 0,
                       value ↦ Nil]]],
            rollback ↦ [
                proposal ↦ 0,
                revision ↦ 0,
                values   ↦ [
                    p ∈ {}  ↦ [
                       index ↦ 0,
                       value ↦ Nil]]]]]
    ∧ proposal = [
          i ∈ {} ↦ [
            transaction ↦ 0,
            commit      ↦ Nil,
            apply       ↦ Nil]]
    ∧ configuration = [
          state  ↦ Pending,
          term   ↦ 0,
          committed ↦ [
              index    ↦ 0,
              revision ↦ 0,
              values   ↦ [
                  p ∈ {}  ↦ [
                      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 ∈ DOMAIN transaction :
          RollbackChange(i)
    ∨ ∃ n ∈ Node :
          ∃ i ∈ DOMAIN transaction :
            ReconcileTransaction(n, i)
    ∨ ∃ n ∈ Node :
          ReconcileConfiguration(n)
    ∨ ∃ n ∈ Node :
          ReconcileMastership(n)
    ∨ ∃ n ∈ Node :
          ∨ ConnectNode(n)
          ∨ DisconnectNode(n)
    ∨ StartTarget
    $\lor\ StopTarget$

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

$Alias\ \triangleq\ [$
    $log \mapsto [$
          $i \in \textsc{domain}\ transaction \mapsto [$
            $change \mapsto$
               $\textsc{if}\ transaction[i].change.proposal \neq 0\ \textsc{then}$
                  $[commit \mapsto proposal[transaction[i].change.proposal].commit,$
                   $apply\ \ \mapsto proposal[transaction[i].change.proposal].apply,$
                   $values\ \mapsto transaction[i].change.values]$
                $\textsc{else}$
                  $[commit \mapsto Nil,$
                   $apply\ \ \mapsto Nil,$
                   $values\ \mapsto transaction[i].change.values],$
            $rollback \mapsto$
               $\textsc{if}\ transaction[i].rollback.proposal \neq 0\ \textsc{then}$
                  $[commit \mapsto proposal[transaction[i].rollback.proposal].commit,$
                   $apply\ \ \mapsto proposal[transaction[i].rollback.proposal].apply,$
                   $values\ \mapsto transaction[i].rollback.values]$
                $\textsc{else}$
                  $[commit \mapsto Nil,$
                   $apply\ \ \mapsto Nil,$
                   $values\ \mapsto transaction[i].rollback.values]]\ @@$
            $transaction[i]],$
    $transaction\ \ \ \mapsto transaction,$
    $proposal\ \ \ \ \ \ \mapsto proposal,$
    $configuration \mapsto configuration,$
    $mastership\ \ \ \ \mapsto mastership,$
```

```
  $conn \quad\quad\quad \mapsto conn,$
  $target \quad\quad\; \mapsto target,$
  $history \quad\quad \mapsto history]$
```

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

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

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

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

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

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

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

```
            $\land j < i$
            $\land history[j].type = Change$
            $\land history[j].phase = p$
            $\land history[j].index > history[i].index$
            $\land \neg\exists k \in \textsc{domain}\ history :$
                $\land k > j$
                $\land k < i$
                $\land history[k].type = Rollback$
                $\land history[k].phase = p$
                $\land history[k].index = history[j].index$

$Order \triangleq$
  $\land\ \forall i \in \textsc{domain}\ history :$
      $\lor IsOrderedChange(Commit, i)$
      $\lor IsOrderedChange(Apply, i)$
      $\lor IsOrderedRollback(Commit, i)$
      $\lor IsOrderedRollback(Apply, i)$
  $\land\ \forall i \in \textsc{domain}\ transaction :$
        $\land transaction[i].change.proposal \neq 0$
        $\land proposal[transaction[i].change.proposal].apply = Failed$
        $\land transaction[i].rollback.proposal \neq 0 \Rightarrow$
            $proposal[transaction[i].rollback.proposal].apply \neq Complete$
        $\Rightarrow \forall j \in \textsc{domain}\ transaction : (j > i \Rightarrow$
            $(transaction[j].change.proposal \neq 0 \Rightarrow$
                $proposal[transaction[j].change.proposal].apply \neq Complete))$

$\textsc{local}\ IsChangeCommitted(i) \triangleq$
  $\land\quad transaction[i].change.proposal \neq 0$
  $\land\quad proposal[transaction[i].change.proposal].commit = Complete$
  $\land\quad transaction[i].rollback.proposal \neq 0 \Rightarrow$
        $proposal[transaction[i].rollback.proposal].commit \neq Complete$

$\textsc{local}\ IsChangeApplied(i) \triangleq$
  $\land\quad transaction[i].change.proposal \neq 0$
  $\land\quad proposal[transaction[i].change.proposal].apply = Complete$
  $\land\quad transaction[i].rollback.proposal \neq 0 \Rightarrow$
        $proposal[transaction[i].rollback.proposal].apply \neq Complete$

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

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

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

$Terminates(i) \triangleq$
  $\land i \in \textsc{domain}\ transaction \land transaction[i].phase = Change \leadsto$
      $\land i \in \textsc{domain}\ transaction$
      $\land transaction[i].change.proposal \neq 0$
      $\land proposal[transaction[i].change.proposal].commit \in Done$
      $\land proposal[transaction[i].change.proposal].apply \in Done$
  $\land i \in \textsc{domain}\ transaction \land transaction[i].phase = Rollback \leadsto$
      $\land i \in \textsc{domain}\ transaction$
      $\land transaction[i].rollback.proposal \neq 0$
      $\land proposal[transaction[i].rollback.proposal].commit \in Done$
      $\land proposal[transaction[i].rollback.proposal].apply \in Done$

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

$Liveness \triangleq Termination$

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