```
──────────────── MODULE ConfigImpl ────────────────
INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

LOCAL INSTANCE TLC
───────────────────────────────────────────────────
```

This section specifies constant parameters for the model.

CONSTANT *LogEnabled*

ASSUME $LogEnabled \in$ BOOLEAN

CONSTANT *None*

ASSUME $None \in$ STRING

CONSTANT *Node*

ASSUME $\forall\, n \in Node : n \in$ STRING

```
CONSTANTS
    Change,
    Rollback
```

$Event \triangleq \{Change, Rollback\}$

ASSUME $\forall\, e \in Event : e \in$ STRING

```
CONSTANTS
    Commit,
    Apply
```

$Phase \triangleq \{Commit, Apply\}$

ASSUME $\forall\, p \in Phase : p \in$ STRING

```
CONSTANTS
    Pending,
    InProgress,
    Complete,
    Aborted,
    Failed
```

$State \triangleq \{Pending, InProgress, Complete, Aborted, Failed\}$

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

ASSUME $\forall\, s \in \mathit{State} : s \in \text{STRING}$

CONSTANT Path

ASSUME $\forall\, p \in \mathit{Path} : p \in \text{STRING}$

CONSTANT Value

ASSUME $\forall\, v \in \mathit{Value} : v \in \text{STRING}$

$\mathit{AllValues} \triangleq \mathit{Value} \cup \{\mathit{None}\}$

CONSTANT $\mathit{NumProposals}$

ASSUME $\mathit{NumProposals} \in \mathit{Nat}$

This section defines model state variables.

```
$proposal \triangleq [\, i \in 1 \,..\, Nat \mapsto [$
    $phase \mapsto Phase,$
    $change \mapsto [$
      $values \mapsto Change,$
      $commit \mapsto State,$
      $apply \mapsto State],$
    $rollback \mapsto [$
      $index \mapsto Nat,$
      $values \mapsto Change,$
      $commit \mapsto State,$
      $apply \mapsto State]]]$
$configuration \triangleq [$
  $committed \mapsto [$
    $index \mapsto Nat,$
    $values \mapsto Change],$
  $applied \mapsto [$
    $index \mapsto Nat,$
    $values \mapsto Change,$
    $term \mapsto Nat]]$
$mastership \triangleq [$
  $master \mapsto \text{STRING},$
  $term \mapsto Nat,$
  $conn \mapsto Nat]$
$conn \triangleq [\, n \in Node \mapsto [$
    $id \qquad\;\; \mapsto Nat,$
    $connected \mapsto \text{BOOLEAN}\ ]]$
$target \triangleq [$
  $id \qquad \mapsto Nat,$
  $values \mapsto Change,$
  $running \mapsto \text{BOOLEAN}\ ]$
```

```
VARIABLE proposal

VARIABLE configuration

VARIABLE mastership

VARIABLE conn

VARIABLE target

VARIABLE history

VARIABLE mapping

vars ≜ ⟨proposal, configuration, mastership, conn, target, history, mapping⟩
```

```
LOCAL MastershipLog ≜ INSTANCE Log WITH
   File       ← "Mastership.log",
   CurrState ← [
      target        ↦ target,
      mastership    ↦ mastership,
      conns         ↦ conn],
   SuccState ← [
      target        ↦ target′,
      mastership    ↦ mastership′,
      conns         ↦ conn′],
   Enabled    ← LogEnabled

LOCAL ConfigurationLog ≜ INSTANCE Log WITH
   File       ← "Configuration.log",
   CurrState ← [
      configuration ↦ configuration,
      target        ↦ target,
      mastership    ↦ mastership,
      conns         ↦ conn],
   SuccState ← [
      configuration ↦ configuration′,
      target        ↦ target′,
      mastership    ↦ mastership′,
      conns         ↦ conn′],
   Enabled    ← LogEnabled

LOCAL ProposalLog ≜ INSTANCE Log WITH
   File       ← "Proposal.log",
   CurrState ← [
      proposals     ↦ [i ∈ {i ∈ DOMAIN proposal : proposal[i].phase ≠ None} ↦ proposal[i]],
      configuration ↦ configuration,
      target        ↦ target,
      mastership    ↦ mastership,
      conns         ↦ conn],
   SuccState ← [
      proposals     ↦ [i ∈ {i ∈ DOMAIN proposal' : proposal'[i].phase ≠ None} ↦ proposal'[i]],
      configuration ↦ configuration',
      target        ↦ target',
      mastership    ↦ mastership',
      conns         ↦ conn'],
   Enabled   ← LogEnabled
```

This section models configuration target.

```
StartTarget ≜
    ∧ ¬target.running
    ∧ target' = [target EXCEPT !.id      = target.id + 1,
                               !.running = TRUE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, conn, history⟩

StopTarget ≜
    ∧ target.running
    ∧ target' = [target EXCEPT !.running = FALSE,
                               !.values  = [p ∈ {} ↦ [value ↦ None]]]
    ∧ conn' = [n ∈ Node ↦ [conn[n] EXCEPT !.connected = FALSE]]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, history⟩
```

This section models nodes connection to the configuration target.

```
ConnectNode(n) ≜
    ∧ ¬conn[n].connected
    ∧ target.running
    ∧ conn' = [conn EXCEPT ![n].id        = conn[n].id + 1,
                           ![n].connected = TRUE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩

DisconnectNode(n) ≜
    ∧ conn[n].connected
    ∧ conn' = [conn EXCEPT ![n].connected = FALSE]
    ∧ UNCHANGED ⟨proposal, configuration, mastership, target, history⟩
```

This section models *mastership* reconciliation.

```
$ReconcileMastership(n) \triangleq$
    $\land \lor \land conn[n].connected$
            $\land mastership.master = None$
            $\land mastership' = [master \mapsto n,\ term \mapsto mastership.term + 1,\ conn \mapsto conn[n].id]$
        $\lor \land \neg conn[n].connected$
            $\land mastership.master = n$
            $\land mastership' = [mastership\ \text{EXCEPT}\ !.master = None]$
    $\land \text{UNCHANGED}\ \langle proposal,\ configuration,\ conn,\ target,\ history \rangle$
```

This section models configuration reconciliation.

```
$ReconcileConfiguration(n) \triangleq$
    $\land mastership.master = n$
    $\land \lor \land configuration.status \neq InProgress$
            $\land configuration.applied.term < mastership.term$
            $\land configuration' = [configuration\ \text{EXCEPT}\ !.status = InProgress]$
            $\land \text{UNCHANGED}\ \langle target \rangle$
        $\lor \land configuration.status = InProgress$
            $\land configuration.applied.term < mastership.term$
            $\land conn[n].connected$
            $\land target.running$
            $\land target' = [target\ \text{EXCEPT}\ !.values = configuration.applied.values]$
            $\land configuration' = [configuration\ \text{EXCEPT}\ !.applied.term = mastership.term,$
                                                        $!.applied.target = target.id,$
                                                        $!.status = Complete]$
    $\land \text{UNCHANGED}\ \langle proposal,\ mastership,\ conn,\ history \rangle$
```

This section models proposal reconcilation.

```
$CommitChange(n,\ i) \triangleq$
    'index' is the current index committed to the configuration
    'changeIndex' is the maximum change index committed to the configuration
    'targetIndex' is the index of the proposal currently being committed
    $targetIndex$ is always changed first. Once the change is committed, the
    $changeIndex$ and index will be incremented to match the $targetIndex$.
    If the index is less than the $targetIndex$, this indicates a rollback
    of a prior proposal is being processed, and the $targetIndex$ cannot be incremented
    until that rollback is complete. The index represents the index to which
    the proposal at $changeIndex + 1$ rolls back.
    $\land \lor \land proposal[i].change.commit = Pending$
            $\land configuration.committed.changeIndex = i - 1$
            $\land \lor \land configuration.committed.targetIndex \neq i$
                    $\land configuration.committed.index = configuration.committed.targetIndex$
                    $\land configuration' = [configuration\ \text{EXCEPT}\ !.committed.targetIndex = i]$
```

```
                $\land$ UNCHANGED $\langle proposal \rangle$
            $\lor \land configuration.committed.targetIndex = i$
                $\land \lor \land proposal[i].rollback.commit = None$
                        $\land$ LET $rollbackIndex \triangleq configuration.committed.index$
                                $rollbackValues \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
                                                IF $p \in$ DOMAIN $configuration.committed.values$ THEN
                                                    $configuration.committed.values[p]$
                                                ELSE
                                                    $[index \mapsto 0, value \mapsto None]]$
                            IN  $\lor \land proposal[i].rollback.commit = None$
                                   $\land proposal' = [proposal$ EXCEPT $![i].change.commit = InProgress,$
                                                                     $![i].rollback.index = rollbackIndex,$
                                                                     $![i].rollback.values = rollbackValues]$
                                $\lor \land proposal[i].rollback.commit = Pending$
                                   $\land proposal' = [proposal$ EXCEPT $![i].change.commit = Aborted,$
                                                                     $![i].rollback.index = rollbackIndex]$
                $\land$ UNCHANGED $\langle configuration \rangle$
        $\land$ UNCHANGED $\langle history \rangle$
    $\lor \land proposal[i].change.commit = InProgress$
        $\land \lor \land configuration.committed.changeIndex = i - 1$
                $\land \lor \land$ LET $values \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
                                        $proposal[i].change.values[p]$ @@ $[index \mapsto i]]$ @@
                                            $configuration.committed.values$
                        IN  $\land configuration' = [configuration$ EXCEPT $!.committed.index = i,$
                                                                          $!.committed.changeIndex = i,$
                                                                          $!.committed.values = values]$
                            $\land history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])$
                            $\land$ UNCHANGED $\langle proposal \rangle$
                    $\lor \land proposal' = [proposal$ EXCEPT $![i].change.commit = Failed]$
                        $\land$ UNCHANGED $\langle configuration, history \rangle$
            $\lor \land configuration.committed.changeIndex \geq i$
                $\land proposal' = [proposal$ EXCEPT $![i].change.commit = Complete]$
                $\land$ UNCHANGED $\langle configuration, history \rangle$
    $\lor \land proposal[i].change.commit \in \{Aborted, Failed\}$
        $\land configuration.committed.changeIndex = i - 1$
        $\land configuration' = [configuration$ EXCEPT $!.committed.index = i,$
                                                      $!.committed.changeIndex = i]$
        $\land$ UNCHANGED $\langle proposal, history \rangle$
    $\land$ UNCHANGED $\langle target \rangle$
```

$ApplyChange(n, i) \triangleq$

'index' is the current index applied to the configuration

'changeIndex' is the maximum change index applied to the configuration

'targetIndex' is the index of the proposal currently being applied

targetIndex is always changed first. Once the change is applied, the

```
$changeIndex$ and index will be incremented to match the $targetIndex$.
If the index is less than the $targetIndex$, this indicates a rollback
of a prior proposal is being processed, and the $targetIndex$ cannot be incremented
until that rollback is complete. The index represents the index to which
the proposal at $changeIndex + 1$ rolls back.
$\land$ $\lor$ $\land$ $proposal[i].change.apply = Pending$
       $\land$ $configuration.committed.changeIndex \geq i$
       $\land$ $configuration.applied.changeIndex = i - 1$
       $\land$ $\lor$ $\land$ $configuration.applied.targetIndex \neq i$
            $\land$ $configuration.applied.index = configuration.applied.targetIndex$
            $\land$ $configuration' = [configuration$ EXCEPT $!.applied.targetIndex = i]$
            $\land$ UNCHANGED $\langle proposal \rangle$
         $\lor$ $\land$ $configuration.applied.targetIndex = i$
            $\land$ $\lor$ $\land$ $proposal[i].change.commit \in \{Aborted, Failed\}$
                 $\land$ $proposal' = [proposal$ EXCEPT $![i].change.apply = Aborted]$
               $\lor$ $\land$ $proposal[i].change.commit = Complete$
                 $\land$ $proposal' = [proposal$ EXCEPT $![i].change.apply = InProgress]$
            $\land$ UNCHANGED $\langle configuration \rangle$
       $\land$ UNCHANGED $\langle target, history \rangle$
   $\lor$ $\land$ $proposal[i].change.apply = InProgress$
       Verify the applied term is the current $mastership$ term to ensure the
       configuration has been synchronized following restarts.
       $\land$ $configuration.applied.term = mastership.term$
       Verify the node's connection to the target.
       $\land$ $conn[n].connected$
       $\land$ $mastership.conn = conn[n].id$
       $\land$ $target.running$
       $\land$ $\lor$ $\land$ $configuration.applied.changeIndex = i - 1$
            $\land$ $\lor$ $\land$ LET $values \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
                                        $proposal[i].change.values[p]$ @@ $[index \mapsto i]]$
                     IN    $\land$ $target' = [target$ EXCEPT $!.values = values$ @@ $target.values]$
                           $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index = i,$
                                                                $!.applied.changeIndex = i,$
                                                                $!.applied.values = values$ @@
                                                                  $configuration.applied.values]$
                           $\land$ $history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$
                           $\land$ UNCHANGED $\langle proposal \rangle$
               $\lor$ $\land$ $proposal' = [proposal$ EXCEPT $![i].change.apply = Failed]$
                 $\land$ UNCHANGED $\langle configuration, target, history \rangle$
         $\lor$ $\land$ $configuration.applied.changeIndex \geq i$
            $\land$ $proposal' = [proposal$ EXCEPT $![i].change.apply = Complete]$
            $\land$ UNCHANGED $\langle configuration, target, history \rangle$
   $\lor$ $\land$ $proposal[i].change.apply = Failed$
       $\land$ $configuration.applied.changeIndex = i - 1$
       $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index            = i,$
```

```
                                                    !.applied.changeIndex = i]
        ∧ UNCHANGED ⟨proposal, target, history⟩

CommitRollback(n, i) ≜
    'index' is the current index committed to the configuration
    'changeIndex' is the maximum change index committed to the configuration
    'targetIndex' is the index of the proposal currently being committed
    targetIndex is always changed first. Once the rollback is committed, the
    index will be decremented to match the targetIndex. The next time a change
    is committed, the index will increase again. If the committed index is equal
    to this proposal index, this proposal is the next to be rolled back. To roll
    back a proposal, the target index is set to the proposal's rollback index.
    When the rollback is committed, the committed index is set to the proposal's
    rollback index, thus matching the targetIndex. This unblocks new changes
    to be committed.
  ∧ ∨ ∧ proposal[i].rollback.commit = Pending
      ∧ configuration.committed.changeIndex ≥ i
      ∧ configuration.committed.index = i
      ∧ ∨ ∧ configuration.committed.targetIndex = i
          ∧ configuration′ = [configuration EXCEPT !.committed.targetIndex = proposal[i].rollback.index]
          ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ configuration.committed.targetIndex = proposal[i].rollback.index
          ∧ ∨ ∧ proposal[i].change.commit ≠ Aborted
                ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = InProgress]
            ∨ ∧ proposal[i].change.commit = Aborted
                ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = Complete]
          ∧ UNCHANGED ⟨configuration⟩
      ∧ UNCHANGED ⟨history⟩
    ∨ ∧ proposal[i].rollback.commit = InProgress
      ∧ ∨ ∧ configuration.committed.index = i
          ∧ LET values ≜ [p ∈ DOMAIN configuration.committed.values ↦
                              IF p ∈ DOMAIN proposal[i].rollback.values THEN
                                 proposal[i].rollback.values[p]
                               ELSE
                                 configuration.committed.values[p]]
            IN   configuration′ = [configuration EXCEPT !.committed.index  = proposal[i].rollback.index,
                                                        !.committed.values = values]
          ∧ history′ = Append(history, [type ↦ Rollback, phase ↦ Commit, index ↦ i])
          ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ configuration.committed.index = proposal[i].rollback.index
          ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = Complete]
          ∧ UNCHANGED ⟨configuration, history⟩
    ∨ ∧ proposal[i].rollback.commit = Complete
      ∧ configuration.committed.targetIndex = proposal[i].rollback.index
      ∧ configuration.committed.index ≠ proposal[i].rollback.index
```

```
            $\land$ $configuration' = [configuration$ EXCEPT $!.committed.index = proposal[i].rollback.index]$
            $\land$ UNCHANGED $\langle proposal, history \rangle$
        $\land$ UNCHANGED $\langle target \rangle$

$ApplyRollback(n, i) \triangleq$
    'index' is the current index applied to the configuration
    'changeIndex' is the maximum change index applied to the configuration
    'targetIndex' is the index of the proposal currently being applied
    targetIndex is always changed first. Once the rollback is applied, the
    index will be decremented to match the targetIndex. The next time a change
    is applied, the index will increase again. If the applied index is equal
    to this proposal index, this proposal is the next to be rolled back. To roll
    back a proposal, the target index is set to the proposal's rollback index.
    When the rollback is applied, the applied index is set to the proposal's
    rollback index, thus matching the targetIndex. This unblocks new changes
    to be applied.
    $\land$ $\lor$ $\land$ $proposal[i].rollback.apply = Pending$
            $\land$ $configuration.committed.index \leq proposal[i].rollback.index$
            $\land$ $configuration.applied.changeIndex \geq i$
            $\land$ $configuration.applied.index = i$
            $\land$ $\lor$ $\land$ $configuration.applied.targetIndex = i$
                  $\land$ $configuration' = [configuration$ EXCEPT $!.applied.targetIndex = proposal[i].rollback.index]$
                  $\land$ UNCHANGED $\langle proposal \rangle$
               $\lor$ $\land$ $configuration.applied.targetIndex = proposal[i].rollback.index$
                  $\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.apply = InProgress]$
                  $\land$ UNCHANGED $\langle configuration \rangle$
            $\land$ UNCHANGED $\langle target, history \rangle$
       $\lor$ $\land$ $proposal[i].rollback.apply = InProgress$
            $\land$ $\lor$ $\land$ $configuration.applied.index = i$
                    Verify the applied term is the current mastership term to ensure the
                    configuration has been synchronized following restarts.
                  $\land$ $configuration.applied.term = mastership.term$
                    Verify the node's connection to the target.
                  $\land$ $conn[n].connected$
                  $\land$ $target.running$
                  $\land$ LET $values \triangleq [p \in$ DOMAIN $configuration.applied.values \mapsto$
                                       IF $p \in$ DOMAIN $proposal[i].rollback.values$ THEN
                                          $proposal[i].rollback.values[p]$
                                        ELSE
                                          $configuration.applied.values[p]]$
                    IN    $\land$ $target' = [target$ EXCEPT $!.values = proposal[i].rollback.values$ @@ $target.values]$
                          $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index = proposal[i].rollback.index,$
                                                                  $!.applied.values = values]$
                  $\land$ $history' = Append(history, [type \mapsto Rollback, phase \mapsto Apply, index \mapsto i])$
                  $\land$ UNCHANGED $\langle proposal \rangle$
            $\lor \land configuration.applied.index \neq i$
               $\land proposal' = [proposal \text{ EXCEPT } ![i].rollback.apply = Complete]$
               $\land \text{UNCHANGED } \langle configuration, target, history \rangle$

$ReconcileProposal(n, i) \triangleq$
    $\land mastership.master = n$
    $\land \lor CommitChange(n, i)$
       $\lor ApplyChange(n, i)$
       $\lor CommitRollback(n, i)$
       $\lor ApplyRollback(n, i)$
    $\land \text{UNCHANGED } \langle mastership, conn \rangle$
```

This section models changes to the proposal queue.

```
 Propose change at index 'i'
$ProposeChange(i) \triangleq$
    $\land proposal[i].phase = None$
    $\land i - 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].phase \neq None$
    $\land \exists p \in Path, v \in AllValues :$
         $\land proposal' = [proposal \text{ EXCEPT } ![i].phase = Change,$
                                              $![i].change.values = (p :> [value \mapsto v]),$
                                              $![i].change.commit = Pending,$
                                              $![i].change.apply = Pending]$
    $\land \text{UNCHANGED } \langle configuration, mastership, conn, target, history \rangle$

 Rollback proposed change at index 'i'
$ProposeRollback(i) \triangleq$
    $\land proposal[i].phase = Change$
    $\land proposal' = [proposal \text{ EXCEPT } ![i].phase = Rollback,$
                                         $![i].rollback.commit = Pending,$
                                         $![i].rollback.apply = Pending]$
    $\land \text{UNCHANGED } \langle configuration, mastership, conn, target, history \rangle$
```

Formal specification, constraints, and theorems.

```
$Init \triangleq$
    $\land proposal = [$
         $i \in 1 \,..\, NumProposals \mapsto [$
           $phase \mapsto None,$
           $change \mapsto [$
               $values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto None]],$
               $commit \mapsto None,$
               $apply \mapsto None],$
           $rollback \mapsto [$
```

```
                $index \mapsto 0,$
                $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
                $commit \mapsto None,$
                $apply \mapsto None]]]$
$\land\ configuration = [$
        $committed \mapsto [$
            $index \mapsto 0,$
            $changeIndex \mapsto 0,$
            $targetIndex \mapsto 0,$
            $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
        $applied \mapsto [$
            $index \mapsto 0,$
            $changeIndex \mapsto 0,$
            $targetIndex \mapsto 0,$
            $term \mapsto 0,$
            $target \mapsto 0,$
            $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
        $status \mapsto Pending]$
$\land\ mastership = [master \mapsto None,\ term \mapsto 0,\ conn \mapsto 0]$
$\land\ conn = [n \in Node \mapsto [id \mapsto 0,\ connected \mapsto \text{FALSE}]]$
$\land\ target = [$
        $id \mapsto 0,$
        $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
        $running \mapsto \text{FALSE}]$
$\land\ history = \langle\rangle$
$\land\ mapping = [$
        $configuration \mapsto [$
                $committed \mapsto [$
                    $values \mapsto configuration.committed.values],$
                $applied \mapsto [$
                    $term \mapsto configuration.applied.term,$
                    $target \mapsto configuration.applied.target,$
                    $values \mapsto configuration.applied.values],$
                $status \mapsto configuration.status],$
        $proposal \mapsto [i \in \text{DOMAIN}\ proposal \mapsto [$
            $phase \mapsto proposal[i].phase,$
            $values \mapsto [p \in \text{DOMAIN}\ proposal[i].change.values \mapsto proposal[i].change.values[p].value],$
            $change \mapsto [$
                $commit \mapsto$ IF $\land\ proposal[i].change.commit = InProgress$
                                  $\land\ configuration.committed.changeIndex \geq i$
                               THEN $Complete$
                               ELSE $proposal[i].change.commit,$
                $apply \mapsto$ IF $\land\ proposal[i].change.apply = InProgress$
                                 $\land\ configuration.applied.changeIndex \geq i$
                                THEN $Complete$
```

```
                          ELSE $proposal[i].change.apply$],
          $rollback \mapsto$ [
              $commit \mapsto$ IF $\land\ proposal[i].rollback.commit\ \ = InProgress$
                                  $\land\ configuration.committed.index \neq i$
                             THEN $Complete$
                             ELSE $proposal[i].rollback.commit$,
              $apply\ \ \ \mapsto$ IF $\land\ proposal[i].rollback.commit = InProgress$
                                  $\land\ configuration.applied.index \neq i$
                             THEN $Complete$
                             ELSE $proposal[i].rollback.apply$]]]]

$Next \triangleq$
   $\land\ \lor\ \exists\, i \in 1\,..\,NumProposals :$
            $\lor\ ProposeChange(i)$
            $\lor\ ProposeRollback(i)$
      $\lor\ \exists\, n \in Node,\ i \in$ DOMAIN $proposal :$
         $ProposalLog!Action(ReconcileProposal(n,\ i),\ [node \mapsto n,\ index \mapsto i])$
      $\lor\ \exists\, n \in Node :$
         $ConfigurationLog!Action(ReconcileConfiguration(n),\ [node \mapsto n])$
      $\lor\ \exists\, n \in Node :$
         $MastershipLog!Action(ReconcileMastership(n),\ [node \mapsto n])$
      $\lor\ \exists\, n \in Node :$
        $\lor\ ConnectNode(n)$
        $\lor\ DisconnectNode(n)$
      $\lor\ StartTarget$
      $\lor\ StopTarget$
   $\land\ mapping' =$ [
         $configuration \mapsto$ [
              $committed \mapsto$ [
                 $values\ \ \mapsto configuration'.committed.values$],
              $applied \mapsto$ [
                 $term\ \ \ \mapsto configuration'.applied.term$,
                 $target\ \mapsto configuration'.applied.target$,
                 $values\ \mapsto configuration'.applied.values$],
              $status \mapsto configuration'.status$],
         $proposal \mapsto [i \in$ DOMAIN $proposal' \mapsto$ [
            $phase\ \ \ \ \mapsto proposal'[i].phase$,
            $values\ \ \ \mapsto [p \in$ DOMAIN $proposal'[i].change.values \mapsto proposal'[i].change.values[p].value]$,
            $change\ \ \ \mapsto$ [
              $commit \mapsto$ IF $\land\ proposal'[i].change.commit = InProgress$
                                 $\land\ configuration'.committed.changeIndex \geq i$
                             THEN $Complete$
                             ELSE $proposal'[i].change.commit$,
              $apply\ \ \ \mapsto$ IF $\land\ proposal'[i].change.apply = InProgress$
                                 $\land\ configuration'.applied.changeIndex \geq i$
```

```
                         $\text{THEN } \mathit{Complete}$
                         $\text{ELSE } \mathit{proposal}'[i].\mathit{change}.\mathit{apply}],$
        $\mathit{rollback} \mapsto [$
            $\mathit{commit} \mapsto \text{IF } \land \mathit{proposal}'[i].\mathit{rollback}.\mathit{commit} = \mathit{InProgress}$
                                 $\land \mathit{configuration}'.\mathit{committed}.\mathit{index} \neq i$
                         $\text{THEN } \mathit{Complete}$
                         $\text{ELSE } \mathit{proposal}'[i].\mathit{rollback}.\mathit{commit},$
            $\mathit{apply} \;\; \mapsto \text{IF } \land \mathit{proposal}'[i].\mathit{rollback}.\mathit{apply} = \mathit{InProgress}$
                                 $\land \mathit{configuration}'.\mathit{applied}.\mathit{index} \neq i$
                         $\text{THEN } \mathit{Complete}$
                         $\text{ELSE } \mathit{proposal}'[i].\mathit{rollback}.\mathit{apply}]]]]$

$\mathit{Spec} \triangleq$
    $\land \mathit{Init}$
    $\land \Box[\mathit{Next}]_{vars}$
    $\land \forall i \in 1 \mathrel{..} \mathit{NumProposals} : \mathrm{WF}_{\langle proposal, configuration, mastership, conn, target, history\rangle}(\mathit{ProposeChange}(i) \lor \mathit{Propo}$
    $\land \forall n \in \mathit{Node}, i \in 1 \mathrel{..} \mathit{NumProposals} : \mathrm{WF}_{\langle proposal, configuration, mastership, conn, target, history\rangle}(\mathit{ReconcilePropos}$
    $\land \forall n \in \mathit{Node} : \mathrm{WF}_{\langle configuration, mastership, conn, target\rangle}(\mathit{ReconcileConfiguration}(n))$
    $\land \forall n \in \mathit{Node} : \mathrm{WF}_{\langle mastership, conn, target\rangle}(\mathit{ReconcileMastership}(n))$
    $\land \forall n \in \mathit{Node} : \mathrm{WF}_{\langle conn, target\rangle}(\mathit{ConnectNode}(n) \lor \mathit{DisconnectNode}(n))$
    $\land \mathrm{WF}_{\langle target\rangle}(\mathit{StartTarget})$
    $\land \mathrm{WF}_{\langle target\rangle}(\mathit{StopTarget})$

$\mathit{Mapping} \triangleq \text{INSTANCE } \mathit{Config} \text{ WITH}$
    $\mathit{proposal} \quad\;\; \leftarrow \mathit{mapping}.\mathit{proposal},$
    $\mathit{configuration} \leftarrow \mathit{mapping}.\mathit{configuration}$

$\mathit{Refinement} \triangleq \mathit{Mapping}!\mathit{Spec}$

$\mathit{Order} \triangleq \mathit{Mapping}!\mathit{Order}$

$\mathit{Consistency} \triangleq \mathit{Mapping}!\mathit{Consistency}$

$\mathit{Liveness} \triangleq \mathit{Mapping}!\mathit{Liveness}$
```