MODULE *ConfigImpl*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

This section specifies constant parameters for the model.

CONSTANT *LogEnabled*

ASSUME $LogEnabled \in \text{BOOLEAN}$

CONSTANT *None*

ASSUME $None \in \text{STRING}$

CONSTANT *Node*

ASSUME $\forall\, n \in Node : n \in \text{STRING}$

CONSTANTS
  Change,
  Rollback

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

ASSUME $\forall\, e \in Event : e \in \text{STRING}$

CONSTANTS
  Commit,
  Apply

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

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

CONSTANTS
  Pending,
  InProgress,
  Complete,
  Aborted,
  Failed

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

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

```
ASSUME ∀ s ∈ State : s ∈ STRING

CONSTANT Path

ASSUME ∀ p ∈ Path : p ∈ STRING

CONSTANT Value

ASSUME ∀ v ∈ Value : v ∈ STRING

AllValues ≜ Value ∪ {None}

CONSTANT NumProposals

ASSUME NumProposals ∈ Nat
```

This section defines model state variables.

```
proposal ≜ [ i ∈ 1 . . Nat ↦ [
    phase ↦ Phase,
    change ↦ [
      values ↦ Change,
      commit ↦ State,
      apply ↦ State],
    rollback ↦ [
      index ↦ Nat,
      values ↦ Change,
      commit ↦ State,
      apply ↦ State]]]

configuration ≜ [
  committed ↦ [
    index ↦ Nat,
    values ↦ Change],
  applied ↦ [
    index ↦ Nat,
    values ↦ Change,
    term ↦ Nat]]

mastership ≜ [
  master ↦ STRING,
  term ↦ Nat,
  conn ↦ Nat]

conn ≜ [ n ∈ Node ↦ [
    id        ↦ Nat,
    connected ↦ BOOLEAN ]]

target ≜ [
  id      ↦ Nat,
  values  ↦ Change,
  running ↦ 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]],
```

$$
\begin{array}{l}
\quad\quad \mathit{configuration} \mapsto \mathit{configuration}, \\
\quad\quad \mathit{target} \quad\quad\;\; \mapsto \mathit{target}, \\
\quad\quad \mathit{mastership} \quad \mapsto \mathit{mastership}, \\
\quad\quad \mathit{conns} \quad\quad\;\; \mapsto \mathit{conn}], \\
\quad \mathit{SuccState} \leftarrow [\\
\quad\quad \mathit{proposals} \quad\;\; \mapsto [i \in \{i \in \text{DOMAIN } \mathit{proposal}' : \mathit{proposal}'[i].\mathit{phase} \neq \mathit{None}\} \mapsto \mathit{proposal}'[i]], \\
\quad\quad \mathit{configuration} \mapsto \mathit{configuration}', \\
\quad\quad \mathit{target} \quad\quad\;\; \mapsto \mathit{target}', \\
\quad\quad \mathit{mastership} \quad \mapsto \mathit{mastership}', \\
\quad\quad \mathit{conns} \quad\quad\;\; \mapsto \mathit{conn}'], \\
\quad \mathit{Enabled} \quad \leftarrow \mathit{LogEnabled}
\end{array}
$$

This section models configuration target.

$$
\begin{array}{l}
\mathit{StartTarget} \triangleq \\
\quad \land \neg \mathit{target}.\mathit{running} \\
\quad \land \mathit{target}' = [\mathit{target} \text{ EXCEPT } !.\mathit{id} \quad\quad\;\; = \mathit{target}.\mathit{id} + 1, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\;\; !.\mathit{running} = \text{TRUE}] \\
\quad \land \text{UNCHANGED } \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{conn}, \mathit{history} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{StopTarget} \triangleq \\
\quad \land \mathit{target}.\mathit{running} \\
\quad \land \mathit{target}' = [\mathit{target} \text{ EXCEPT } !.\mathit{running} = \text{FALSE}, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\;\; !.\mathit{values} \quad = [p \in \{\} \mapsto [\mathit{value} \mapsto \mathit{None}]]] \\
\quad \land \mathit{conn}' = [n \in \mathit{Node} \mapsto [\mathit{conn}[n] \text{ EXCEPT } !.\mathit{connected} = \text{FALSE}]] \\
\quad \land \text{UNCHANGED } \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{history} \rangle
\end{array}
$$

This section models nodes connection to the configuration target.

$$
\begin{array}{l}
\mathit{ConnectNode}(n) \triangleq \\
\quad \land \neg \mathit{conn}[n].\mathit{connected} \\
\quad \land \mathit{target}.\mathit{running} \\
\quad \land \mathit{conn}' = [\mathit{conn} \text{ EXCEPT } ![n].\mathit{id} \quad\quad\quad = \mathit{conn}[n].\mathit{id} + 1, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\; ![n].\mathit{connected} = \text{TRUE}] \\
\quad \land \text{UNCHANGED } \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{target}, \mathit{history} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{DisconnectNode}(n) \triangleq \\
\quad \land \mathit{conn}[n].\mathit{connected} \\
\quad \land \mathit{conn}' = [\mathit{conn} \text{ EXCEPT } ![n].\mathit{connected} = \text{FALSE}] \\
\quad \land \text{UNCHANGED } \langle \mathit{proposal}, \mathit{configuration}, \mathit{mastership}, \mathit{target}, \mathit{history} \rangle
\end{array}
$$

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.

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

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.
    $\land \lor \land proposal[i].change.commit = Pending$
            $\land configuration.committed.changeIndex = i - 1$
            $\land \lor \land configuration.committed.targetIndex \neq i$
                    $\land configuration.committed.index = configuration.committed.targetIndex$
                    $\land configuration' = [configuration \text{ EXCEPT } !.committed.targetIndex = i]$
```

```
          ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ configuration.committed.targetIndex = i
          ∧ ∨ ∧ proposal[i].rollback.commit = None
               ∧ LET rollbackIndex  ≜ configuration.committed.index
                      rollbackValues ≜ [p ∈ DOMAIN proposal[i].change.values ↦
                                          IF p ∈ DOMAIN configuration.committed.values THEN
                                             configuration.committed.values[p]
                                           ELSE
                                             [index ↦ 0, value ↦ None]]
                 IN   ∨ ∧ proposal[i].rollback.commit = None
                         ∧ proposal' = [proposal EXCEPT ![i].change.commit   = InProgress,
                                                        ![i].rollback.index   = rollbackIndex,
                                                        ![i].rollback.values  = rollbackValues]
                      ∨ ∧ proposal[i].rollback.commit = Pending
                         ∧ proposal' = [proposal EXCEPT ![i].change.commit  = Aborted,
                                                        ![i].rollback.index  = rollbackIndex]
          ∧ UNCHANGED ⟨configuration⟩
     ∧ UNCHANGED ⟨history⟩
  ∨ ∧ proposal[i].change.commit = InProgress
     ∧ ∨ ∧ configuration.committed.changeIndex = i − 1
          ∧ ∨ ∧ LET values ≜ [p ∈ DOMAIN proposal[i].change.values ↦
                                    proposal[i].change.values[p] @@ [index ↦ i]] @@
                                      configuration.committed.values
                 IN   ∧ configuration' = [configuration EXCEPT !.committed.index       = i,
                                                               !.committed.changeIndex = i,
                                                               !.committed.values      = values]
                      ∧ history' = Append(history, [type ↦ Change, phase ↦ Commit, index ↦ i])
                      ∧ UNCHANGED ⟨proposal⟩
             ∨ ∧ proposal' = [proposal EXCEPT ![i].change.commit = Failed]
                ∧ UNCHANGED ⟨configuration, history⟩
        ∨ ∧ configuration.committed.changeIndex ≥ i
           ∧ proposal' = [proposal EXCEPT ![i].change.commit = Complete]
           ∧ UNCHANGED ⟨configuration, history⟩
  ∨ ∧ proposal[i].change.commit ∈ {Aborted, Failed}
     ∧ configuration.committed.changeIndex = i − 1
     ∧ configuration' = [configuration EXCEPT !.committed.index       = i,
                                              !.committed.changeIndex = i]
     ∧ UNCHANGED ⟨proposal, history⟩
∧ UNCHANGED ⟨target⟩
```

ApplyChange(*n*, *i*) ≜

'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.
$\land \lor \land proposal[i].change.apply = Pending$
        $\land configuration.committed.changeIndex \geq i$
        $\land configuration.applied.changeIndex = i - 1$
        $\land \lor \land configuration.applied.targetIndex \neq i$
              $\land configuration.applied.index = configuration.applied.targetIndex$
              $\land i - 1 \in \textsc{domain}\ proposal \land proposal[i-1].change.apply = Failed \Rightarrow$
                     $proposal[i-1].rollback.apply = Complete$
              $\land configuration' = [configuration\ \textsc{except}\ !.applied.targetIndex = i]$
              $\land \textsc{unchanged}\ \langle proposal \rangle$
           $\lor \land configuration.applied.targetIndex = i$
              $\land \lor \land proposal[i].change.commit \in \{Aborted, Failed\}$
                    $\land proposal' = [proposal\ \textsc{except}\ ![i].change.apply = Aborted]$
                 $\lor \land proposal[i].change.commit = Complete$
                    $\land proposal' = [proposal\ \textsc{except}\ ![i].change.apply = InProgress]$
              $\land \textsc{unchanged}\ \langle configuration \rangle$
        $\land \textsc{unchanged}\ \langle target, history \rangle$
   $\lor \land proposal[i].change.apply = InProgress$
        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.
        $\land conn[n].connected$
        $\land mastership.conn = conn[n].id$
        $\land target.running$
        $\land \lor \land configuration.applied.changeIndex = i - 1$
              $\land \lor \land \textsc{let}\ values \triangleq [p \in \textsc{domain}\ proposal[i].change.values \mapsto$
                                      $proposal[i].change.values[p]\ @@\ [index \mapsto i]]$
                       $\textsc{in}\ \land target' = [target\ \textsc{except}\ !.values = values\ @@\ target.values]$
                           $\land configuration' = [configuration\ \textsc{except}\ !.applied.index = i,$
                                                          $!.applied.changeIndex = i,$
                                                          $!.applied.values = values\ @@$
                                                            $configuration.applied.values]$
                           $\land history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$
                           $\land \textsc{unchanged}\ \langle proposal \rangle$
                 $\lor \land proposal' = [proposal\ \textsc{except}\ ![i].change.apply = Failed]$
                    $\land \textsc{unchanged}\ \langle configuration, target, history \rangle$
           $\lor \land configuration.applied.changeIndex \geq i$
              $\land proposal' = [proposal\ \textsc{except}\ ![i].change.apply = Complete]$
              $\land \textsc{unchanged}\ \langle configuration, target, history \rangle$
   $\lor \land proposal[i].change.apply = Failed$
```

```
        $\land configuration.applied.changeIndex = i - 1$
        $\land configuration' = [configuration$ EXCEPT $!.applied.index        = i,$
                                                 $!.applied.changeIndex = i]$
        $\land$ UNCHANGED $\langle proposal, target, history \rangle$

$CommitRollback(n, i) \triangleq$
    '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 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.
    $\land \lor \land proposal[i].rollback.commit = Pending$
            $\land configuration.committed.changeIndex \geq i$
            $\land configuration.committed.index = i$
            $\land \lor \land configuration.committed.targetIndex = i$
                  $\land configuration' = [configuration$ EXCEPT $!.committed.targetIndex = proposal[i].rollback.index]$
                  $\land$ UNCHANGED $\langle proposal \rangle$
               $\lor \land configuration.committed.targetIndex = proposal[i].rollback.index$
                  $\land \lor \land proposal[i].change.commit \neq Aborted$
                        $\land proposal' = [proposal$ EXCEPT $![i].rollback.commit = InProgress]$
                     $\lor \land proposal[i].change.commit = Aborted$
                        $\land proposal' = [proposal$ EXCEPT $![i].rollback.commit = Complete]$
                  $\land$ UNCHANGED $\langle configuration \rangle$
            $\land$ UNCHANGED $\langle history \rangle$
         $\lor \land proposal[i].rollback.commit = InProgress$
            $\land \lor \land configuration.committed.index = i$
                  $\land configuration' = [configuration$ EXCEPT $!.committed.index  = proposal[i].rollback.index,$
                                                           $!.committed.values = proposal[i].rollback.values$ @@
                                                             $configuration.committed.values]$
                  $\land history' = Append(history, [type \mapsto Rollback, phase \mapsto Commit, index \mapsto i])$
                  $\land$ UNCHANGED $\langle proposal \rangle$
               $\lor \land configuration.committed.index = proposal[i].rollback.index$
                  $\land proposal' = [proposal$ EXCEPT $![i].rollback.commit = Complete]$
                  $\land$ UNCHANGED $\langle configuration, history \rangle$
         $\lor \land proposal[i].rollback.commit = Complete$
            $\land proposal[i].change.commit = Aborted$
            $\land configuration.committed.targetIndex = proposal[i].rollback.index$
            $\land configuration.committed.index \neq proposal[i].rollback.index$
            $\land configuration' = [configuration$ 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.
               $\land$ $conn[n].connected$
               $\land$ $target.running$
               $\land$ $target' = [target$ EXCEPT $!.values = proposal[i].rollback.values$ @@ $target.values]$
               $\land$ $configuration' = [configuration$ EXCEPT $!.applied.index\ \ = proposal[i].rollback.index,$
                                                  $!.applied.values\ \ = proposal[i].rollback.values$ @@
                                                    $configuration.applied.values]$
               $\land$ $history' = Append(history, [type \mapsto Rollback, phase \mapsto Apply, index \mapsto i])$
               $\land$ UNCHANGED $\langle proposal \rangle$
            $\lor$ $\land$ $configuration.applied.index \neq i$
               $\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.apply = Complete]$
               $\land$ UNCHANGED $\langle configuration, target, history \rangle$

$ReconcileProposal(n, i) \triangleq$
```

```
    ∧ mastership.master = n
    ∧ ∨ CommitChange(n, i)
      ∨ ApplyChange(n, i)
      ∨ CommitRollback(n, i)
      ∨ ApplyRollback(n, i)
    ∧ UNCHANGED ⟨mastership, conn⟩
```

This section models changes to the proposal queue.

Propose change at index 'i'

```
ProposeChange(i) ≜
    ∧ proposal[i].phase = None
    ∧ i − 1 ∈ DOMAIN proposal ⇒ proposal[i − 1].phase ≠ None
    ∧ ∃ p ∈ Path, v ∈ AllValues :
        ∧ proposal′ = [proposal EXCEPT ![i].phase          = Change,
                                       ![i].change.values  = (p :> [value ↦ v]),
                                       ![i].change.commit = Pending,
                                       ![i].change.apply  = Pending]
    ∧ UNCHANGED ⟨configuration, mastership, conn, target, history⟩
```

Rollback proposed change at index 'i'

```
ProposeRollback(i) ≜
    ∧ proposal[i].phase = Change
    ∧ proposal′ = [proposal EXCEPT ![i].phase           = Rollback,
                                   ![i].rollback.commit = Pending,
                                   ![i].rollback.apply  = Pending]
    ∧ UNCHANGED ⟨configuration, mastership, conn, target, history⟩
```

Formal specification, constraints, and theorems.

```
Init ≜
    ∧ proposal = [
        i ∈ 1 . . NumProposals ↦ [
          phase    ↦ None,
          change   ↦ [
              values ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ None]],
              commit ↦ None,
              apply  ↦ None],
          rollback ↦ [
              index  ↦ 0,
              values ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ None]],
              commit ↦ None,
              apply  ↦ None]]]
    ∧ configuration = [
        $committed \mapsto [$
            $index \mapsto 0,$
            $changeIndex \mapsto 0,$
            $targetIndex \mapsto 0,$
            $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
        $applied \mapsto [$
            $index \mapsto 0,$
            $changeIndex \mapsto 0,$
            $targetIndex \mapsto 0,$
            $term \mapsto 0,$
            $target \mapsto 0,$
            $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
        $status \mapsto Pending]$
$\land\ mastership = [master \mapsto None,\ term \mapsto 0,\ conn \mapsto 0]$
$\land\ conn = [n \in Node \mapsto [id \mapsto 0,\ connected \mapsto \text{FALSE}]]$
$\land\ target = [$
        $id \mapsto 0,$
        $values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
        $running \mapsto \text{FALSE}]$
$\land\ history = \langle\rangle$
$\land\ mapping = [$
        $configuration \mapsto [$
            $committed \mapsto [$
                $values \mapsto configuration.committed.values],$
            $applied \mapsto [$
                $term \mapsto configuration.applied.term,$
                $target \mapsto configuration.applied.target,$
                $values \mapsto configuration.applied.values],$
            $status \mapsto configuration.status],$
        $proposal \mapsto [i \in \text{DOMAIN}\ proposal \mapsto [$
            $phase \mapsto proposal[i].phase,$
            $values \mapsto [p \in \text{DOMAIN}\ proposal[i].change.values \mapsto proposal[i].change.values[p].value],$
            $change \mapsto [$
                $commit \mapsto \text{IF}\ \land proposal[i].change.commit = InProgress$
                                $\land configuration.committed.changeIndex \geq i$
                            $\text{THEN}\ Complete$
                            $\text{ELSE}\ proposal[i].change.commit,$
                $apply \mapsto \text{IF}\ \land proposal[i].change.apply = InProgress$
                                $\land configuration.applied.changeIndex \geq i$
                            $\text{THEN}\ Complete$
                            $\text{ELSE}\ proposal[i].change.apply],$
            $rollback \mapsto [$
                $commit \mapsto \text{IF}\ \land proposal[i].rollback.commit = InProgress$
                                $\land configuration.committed.index \neq i$
                            $\text{THEN}\ Complete$
```

```
                    $\textsc{else}\ \ proposal[i].rollback.commit,$
          $apply\ \ \ \mapsto \textsc{if}\ \ \land proposal[i].rollback.commit = InProgress$
                       $\land configuration.applied.index \neq i$
                    $\textsc{then}\ \ Complete$
                    $\textsc{else}\ \ proposal[i].rollback.apply]]]]$

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

$$
\begin{array}{lll}
 & & \textsc{then}\ \ \mathit{Complete} \\
 & & \textsc{else}\ \ \mathit{proposal}'[i].\mathit{rollback}.\mathit{commit}, \\
\mathit{apply} & \mapsto & \textsc{if}\ \ \land \mathit{proposal}'[i].\mathit{rollback}.\mathit{apply} = \mathit{InProgress} \\
 & & \qquad \land \mathit{configuration}'.\mathit{applied}.\mathit{index} \neq i \\
 & & \textsc{then}\ \ \mathit{Complete} \\
 & & \textsc{else}\ \ \mathit{proposal}'[i].\mathit{rollback}.\mathit{apply}]]]]
\end{array}
$$

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

$$
\begin{array}{l}
\mathit{Mapping} \triangleq \textsc{instance}\ \mathit{Config}\ \textsc{with} \\
\quad \mathit{proposal} \qquad\ \ \leftarrow \mathit{mapping}.\mathit{proposal}, \\
\quad \mathit{configuration} \leftarrow \mathit{mapping}.\mathit{configuration}
\end{array}
$$

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

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

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

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