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 $\forall\, s \in State : s \in$ STRING

CONSTANT $Path$

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

CONSTANT $Value$

ASSUME $\forall\, v \in Value : v \in$ STRING

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

CONSTANT $NumProposals$

ASSUME $NumProposals \in Nat$

This section defines model state variables.

$proposal \triangleq [\, i \in 1 \,..\, Nat \mapsto [$
$\quad phase \mapsto Phase,$
$\quad change \mapsto [$
$\quad\quad values \mapsto Change,$
$\quad\quad commit \mapsto State,$
$\quad\quad apply \mapsto State],$
$\quad rollback \mapsto [$
$\quad\quad index \mapsto Nat,$
$\quad\quad values \mapsto Change,$
$\quad\quad commit \mapsto State,$
$\quad\quad apply \mapsto State]]]$

$configuration \triangleq [$
$\quad committed \mapsto [$
$\quad\quad index \mapsto Nat,$
$\quad\quad values \mapsto Change],$
$\quad applied \mapsto [$
$\quad\quad index \mapsto Nat,$
$\quad\quad values \mapsto Change,$
$\quad\quad term \mapsto Nat]]$

$mastership \triangleq [$
$\quad master \mapsto$ STRING,
$\quad term \mapsto Nat,$
$\quad conn \mapsto Nat]$

$conn \triangleq [\, n \in Node \mapsto [$
$\quad id \quad\quad \mapsto Nat,$
$\quad connected \mapsto$ BOOLEAN $]]$

$target \triangleq [$
$\quad id \quad\quad \mapsto Nat,$
$\quad values \mapsto Change,$
$\quad running \mapsto$ BOOLEAN $]$

VARIABLE *proposal*

VARIABLE *configuration*

VARIABLE *mastership*

VARIABLE *conn*

VARIABLE *target*

VARIABLE *history*

VARIABLE *mapping*

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

LOCAL *MastershipLog* $\triangleq$ INSTANCE *Log* WITH
  File $\leftarrow$ "Mastership.log",
  CurrState $\leftarrow$ [
    target $\mapsto$ *target*,
    mastership $\mapsto$ *mastership*,
    conns $\mapsto$ *conn*],
  SuccState $\leftarrow$ [
    target $\mapsto$ *target'*,
    mastership $\mapsto$ *mastership'*,
    conns $\mapsto$ *conn'*],
  Enabled $\leftarrow$ *LogEnabled*

LOCAL *ConfigurationLog* $\triangleq$ INSTANCE *Log* WITH
  File $\leftarrow$ "Configuration.log",
  CurrState $\leftarrow$ [
    configuration $\mapsto$ *configuration*,
    target $\mapsto$ *target*,
    mastership $\mapsto$ *mastership*,
    conns $\mapsto$ *conn*],
  SuccState $\leftarrow$ [
    configuration $\mapsto$ *configuration'*,
    target $\mapsto$ *target'*,
    mastership $\mapsto$ *mastership'*,
    conns $\mapsto$ *conn'*],
  Enabled $\leftarrow$ *LogEnabled*

LOCAL *ProposalLog* $\triangleq$ INSTANCE *Log* WITH
  File $\leftarrow$ "Proposal.log",
  CurrState $\leftarrow$ [
    proposals $\mapsto$ $[i \in \{i \in \text{DOMAIN } proposal : proposal[i].phase \neq None\} \mapsto proposal[i]]$,

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

This section models configuration target.

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

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

This section models nodes connection to the configuration target.

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

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

This section models *mastership* reconciliation.

$ReconcileMastership(n) \triangleq$

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

This section models configuration reconciliation.

$ReconcileConfiguration(n) \triangleq$

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

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.

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

```
                    ∧ 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.

$\land \lor \land proposal[i].change.apply = Pending$
$\land configuration.committed.changeIndex \geq i$
$\land configuration.applied.changeIndex = i - 1$
$\land \lor \land configuration.applied.targetIndex \neq i$
$\land configuration.applied.index = configuration.applied.targetIndex$
$\land i - 1 \in \text{DOMAIN } proposal \land proposal[i-1].change.apply = Failed \Rightarrow$
$proposal[i-1].rollback.apply = Complete$
$\land configuration' = [configuration \text{ EXCEPT } !.applied.targetIndex = i]$
$\land \text{UNCHANGED } \langle proposal \rangle$
$\lor \land configuration.applied.targetIndex = i$
$\land \lor \land proposal[i].change.commit \in \{Aborted, Failed\}$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Aborted]$
$\lor \land proposal[i].change.commit = Complete$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = InProgress]$
$\land \text{UNCHANGED } \langle configuration \rangle$
$\land \text{UNCHANGED } \langle target, history \rangle$
$\lor \land proposal[i].change.apply = InProgress$

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.

$\land conn[n].connected$
$\land mastership.conn = conn[n].id$
$\land target.running$
$\land \lor \land configuration.applied.changeIndex = i - 1$
$\land \lor \land \text{LET } values \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
$proposal[i].change.values[p] \mathbin{@@} [index \mapsto i]]$
$\text{IN} \quad \land target' = [target \text{ EXCEPT } !.values = values \mathbin{@@} target.values]$
$\land configuration' = [configuration \text{ EXCEPT } !.applied.index = i,$
$!.applied.changeIndex = i,$
$!.applied.values = values \mathbin{@@}$
$configuration.applied.values]$
$\land history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$
$\land \text{UNCHANGED } \langle proposal \rangle$
$\lor \land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Failed]$
$\land \text{UNCHANGED } \langle configuration, target, history \rangle$
$\lor \land configuration.applied.changeIndex \geq i$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = Complete]$
$\land \text{UNCHANGED } \langle configuration, target, history \rangle$
$\lor \land proposal[i].change.apply = Failed$

$\land$ $configuration.applied.changeIndex = i - 1$
$\land$ $configuration' = [configuration$ 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 $\langle history \rangle$
$\lor \land proposal[i].rollback.commit = InProgress$
$\land \lor \land$ $configuration.committed.index = i$
$\land$ LET $values \triangleq [p \in$ DOMAIN $configuration.committed.values \mapsto$
IF $p \in$ DOMAIN $proposal[i].rollback.values$ THEN
$proposal[i].rollback.values[p]$
ELSE
$configuration.committed.values[p]]$
IN $configuration' = [configuration$ EXCEPT $!.committed.index = proposal[i].rollback.index,$
$!.committed.values = values]$
$\land$ $history' = Append(history, [type \mapsto Rollback, phase \mapsto Commit, index \mapsto i])$
$\land$ 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.
$\land$ $conn[n].connected$
$\land$ $target.running$
$\land$ 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]
            ∧ history' = Append(history, [type ↦ Rollback, phase ↦ Apply, index ↦ i])
            ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ configuration.applied.index ≠ i
            ∧ proposal' = [proposal EXCEPT ![i].rollback.apply = Complete]
            ∧ UNCHANGED ⟨configuration, target, history⟩

ReconcileProposal(n, i) ≜
    ∧ 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]],
```

$$
\begin{array}{l}
\qquad\qquad\quad commit \mapsto None,\\
\qquad\qquad\quad apply \quad \mapsto None],\\
\qquad\qquad rollback \mapsto [\\
\qquad\qquad\quad index \quad \mapsto 0,\\
\qquad\qquad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],\\
\qquad\qquad\quad commit \mapsto None,\\
\qquad\qquad\quad apply \quad \mapsto None]]]\\
\land\ configuration = [\\
\qquad committed \mapsto [\\
\qquad\quad index \qquad\quad \mapsto 0,\\
\qquad\quad changeIndex \mapsto 0,\\
\qquad\quad targetIndex \ \ \mapsto 0,\\
\qquad\quad values \qquad\ \ \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],\\
\qquad applied \mapsto [\\
\qquad\quad index \qquad\quad \mapsto 0,\\
\qquad\quad changeIndex \mapsto 0,\\
\qquad\quad targetIndex \ \ \mapsto 0,\\
\qquad\quad term \qquad\quad\ \mapsto 0,\\
\qquad\quad target \qquad\quad \mapsto 0,\\
\qquad\quad values \qquad\ \ \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],\\
\qquad 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 = [\\
\qquad id \qquad\ \ \mapsto 0,\\
\qquad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],\\
\qquad running \mapsto \text{FALSE}]\\
\land\ history \quad = \langle\rangle\\
\land\ mapping = [\\
\qquad configuration \mapsto [\\
\qquad\qquad committed \mapsto [\\
\qquad\qquad\quad values \quad \mapsto configuration.committed.values],\\
\qquad\qquad applied \mapsto [\\
\qquad\qquad\quad term \quad \mapsto configuration.applied.term,\\
\qquad\qquad\quad target \ \ \mapsto configuration.applied.target,\\
\qquad\qquad\quad values \mapsto configuration.applied.values],\\
\qquad\qquad status \mapsto configuration.status],\\
\qquad proposal \mapsto [i \in \text{DOMAIN}\ proposal \mapsto [\\
\qquad\quad phase \qquad \mapsto proposal[i].phase,\\
\qquad\quad values \quad\ \ \mapsto [p \in \text{DOMAIN}\ proposal[i].change.values \mapsto proposal[i].change.values[p].value],\\
\qquad\quad change \quad \mapsto [\\
\qquad\qquad commit \mapsto \text{IF}\ \ \land\ proposal[i].change.commit = InProgress\\
\qquad\qquad\qquad\qquad\qquad \land\ configuration.committed.changeIndex \geq i\\
\qquad\qquad\qquad\qquad\ \text{THEN}\ \ Complete\\
\qquad\qquad\qquad\qquad\ \text{ELSE}\ \ proposal[i].change.commit,
\end{array}
$$

```
                apply    ↦ IF  ∧ proposal[i].change.apply = InProgress
                               ∧ configuration.applied.changeIndex ≥ i
                               THEN Complete
                               ELSE proposal[i].change.apply],
            rollback ↦ [
                commit ↦ IF  ∧ proposal[i].rollback.commit  = InProgress
                               ∧ configuration.committed.index ≠ i
                             THEN Complete
                             ELSE proposal[i].rollback.commit,
                apply    ↦ IF  ∧ proposal[i].rollback.commit = InProgress
                               ∧ configuration.applied.index ≠ i
                             THEN Complete
                             ELSE proposal[i].rollback.apply]]]]

Next ≜
  ∧ ∨ ∃ i ∈ 1 .. NumProposals :
        ∨ ProposeChange(i)
        ∨ ProposeRollback(i)
    ∨ ∃ n ∈ Node, i ∈ DOMAIN proposal :
        ProposalLog!Action(ReconcileProposal(n, i), [node ↦ n, index ↦ i])
    ∨ ∃ n ∈ Node :
        ConfigurationLog!Action(ReconcileConfiguration(n), [node ↦ n])
    ∨ ∃ n ∈ Node :
        MastershipLog!Action(ReconcileMastership(n), [node ↦ n])
    ∨ ∃ n ∈ Node :
      ∨ ConnectNode(n)
      ∨ DisconnectNode(n)
    ∨ StartTarget
    ∨ StopTarget
  ∧ mapping' = [
        configuration ↦ [
            committed ↦ [
              values   ↦ configuration'.committed.values],
            applied ↦ [
              term   ↦ configuration'.applied.term,
              target ↦ configuration'.applied.target,
              values ↦ configuration'.applied.values],
            status ↦ configuration'.status],
        proposal ↦ [i ∈ DOMAIN proposal' ↦ [
          phase    ↦ proposal'[i].phase,
          values   ↦ [p ∈ DOMAIN proposal'[i].change.values ↦ proposal'[i].change.values[p].value],
          change   ↦ [
            commit ↦ IF  ∧ proposal'[i].change.commit = InProgress
                         ∧ configuration'.committed.changeIndex ≥ i
                       THEN Complete
```

```
                ELSE  proposal'[i].change.commit,
        apply   ↦ IF  ∧ proposal'[i].change.apply = InProgress
                      ∧ configuration'.applied.changeIndex ≥ i
                  THEN Complete
                  ELSE proposal'[i].change.apply],
    rollback ↦ [
        commit ↦ IF  ∧ proposal'[i].rollback.commit = InProgress
                     ∧ configuration'.committed.index ≠ i
                 THEN Complete
                 ELSE proposal'[i].rollback.commit,
        apply   ↦ IF  ∧ proposal'[i].rollback.apply = InProgress
                      ∧ configuration'.applied.index ≠ i
                  THEN Complete
                  ELSE proposal'[i].rollback.apply]]]]

Spec ≜
  ∧ Init
  ∧ □[Next]_vars
  ∧ ∀ i ∈ 1 .. NumProposals : WF_⟨proposal, configuration, mastership, conn, target, history⟩(ProposeChange(i) ∨ Propo
  ∧ ∀ n ∈ Node, i ∈ 1 .. NumProposals : WF_⟨proposal, configuration, mastership, conn, target, history⟩(ReconcilePropos
  ∧ ∀ n ∈ Node : WF_⟨configuration, mastership, conn, target⟩(ReconcileConfiguration(n))
  ∧ ∀ n ∈ Node : WF_⟨mastership, conn, target⟩(ReconcileMastership(n))
  ∧ ∀ n ∈ Node : WF_⟨conn, target⟩(ConnectNode(n) ∨ DisconnectNode(n))
  ∧ WF_⟨target⟩(StartTarget)
  ∧ WF_⟨target⟩(StopTarget)

Mapping ≜ INSTANCE Config WITH
  proposal      ← mapping.proposal,
  configuration ← mapping.configuration

Refinement ≜ Mapping!Spec

Order ≜ Mapping!Order

Consistency ≜ Mapping!Consistency

Liveness ≜ Mapping!Liveness
```