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

INSTANCE FiniteSets

INSTANCE Sequences

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

This section specifies constant parameters for the model.

```
CONSTANT LogEnabled

ASSUME LogEnabled ∈ BOOLEAN

CONSTANT None

ASSUME None ∈ STRING

CONSTANT Node

ASSUME ∀ n ∈ Node : n ∈ STRING

CONSTANTS
   Change,
   Rollback

Event ≜ {Change, Rollback}

ASSUME ∀ e ∈ Event : e ∈ STRING

CONSTANTS
   Commit,
   Apply

Phase ≜ {Commit, Apply}

ASSUME ∀ p ∈ Phase : p ∈ STRING

CONSTANTS
   Pending,
   InProgress,
   Complete,
   Aborted,
   Failed

State ≜ {Pending, InProgress, Complete, Aborted, Failed}

Working ≜ {Pending, InProgress}
```

```
Finished ≜ {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

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

├─────────────────────────────────────────────────────────────────────┤

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.

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

This section models proposal reconcilation.

```
$CommitChange(n, i) \triangleq$
    $\land \lor \land proposal[i].change.commit = Pending$
              To apply a change, the prior change must have been committed. Additionally,
              the configuration's applied index must match the proposed index to prevent
              commits while a prior change is still being rolled back.
             $\land i - 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].change.commit \in Finished$
             $\land proposal[i].rollback.commit = None$
             $\land \lor \land configuration.committed.proposal < i$
                   $\land configuration.committed.index = configuration.committed.proposal$
                   $\land configuration' = [configuration \text{ EXCEPT } !.committed.proposal = i]$
                   $\land \text{UNCHANGED } \langle proposal \rangle$
                $\lor \land configuration.committed.proposal = i$
                   $\land configuration.committed.index \neq i$
                   $\land \lor \text{LET } rollbackIndex \triangleq configuration.committed.index$
                              $rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
```

```
                                          IF $p \in$ DOMAIN $configuration.committed.values$ THEN
                                             $configuration.committed.values[p]$
                                           ELSE
                                             $[index \mapsto 0,\ value \mapsto None]]$
                         IN    $proposal' = [proposal$ EXCEPT $![i].rollback.index\ \ = rollbackIndex,$
                                                           $![i].rollback.values\ = rollbackValues,$
                                                           $![i].change.commit = InProgress]$
                        $\lor\ proposal' = [proposal$ EXCEPT $![i].change.commit = Failed]$
                      $\land$ UNCHANGED $\langle configuration \rangle$
                  $\land$ UNCHANGED $\langle history \rangle$
                $\lor\ \land\ proposal[i].change.commit = InProgress$
                  $\land\ \lor\ \land\ configuration.committed.index \neq configuration.committed.proposal$
                          $\land$ 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 $!.committed.index\ \ = i,$
                                                                              $!.committed.values = values]$
                                  $\land\ history' = Append(history,\ [type \mapsto Change,\ phase \mapsto Commit,\ index \mapsto i])$
                                  $\land$ UNCHANGED $\langle proposal \rangle$
                      $\lor\ \land\ configuration.committed.proposal = i$
                          $\land\ configuration.committed.index = i$
                          $\land\ proposal' = [proposal$ EXCEPT $![i].change.commit = Complete]$
                          $\land$ UNCHANGED $\langle configuration,\ history \rangle$
                $\lor\ \land\ proposal[i].change.commit = Failed$
                    $\land\ configuration.committed.proposal = i$
                    $\land\ configuration.committed.index \neq i$
                    $\land\ configuration' = [configuration$ EXCEPT $!.committed.index = configuration.committed.index]$
                    $\land$ UNCHANGED $\langle proposal,\ history \rangle$
              $\land$ UNCHANGED $\langle mastership,\ conn,\ target \rangle$

$ApplyChange(n,\ i) \triangleq$
  $\land\ \lor\ \land\ proposal[i].change.apply = Pending$
          $\land\ proposal[i].rollback.apply = None$
          $\land\ \lor\ \land\ proposal[i].change.commit = Complete$
                  $\land\ \forall j \in$ DOMAIN $proposal : j < i \Rightarrow$
                      $\lor\ \land\ proposal[j].change.apply = Complete$
                          $\land\ proposal[j].rollback.apply \notin Working$
                      $\lor\ \land\ proposal[j].change.apply = Failed$
                          $\land\ proposal[j].rollback.apply = Complete$
                  $\land\ proposal' = [proposal$ EXCEPT $![i].change.apply = InProgress]$
              $\lor\ \land\ proposal[i].change.commit \in \{Aborted,\ Failed\}$
                  $\land\ proposal' = [proposal$ EXCEPT $![i].change.apply = Aborted]$
          $\land$ UNCHANGED $\langle configuration,\ target,\ history \rangle$
      $\lor\ \land\ proposal[i].change.apply = InProgress$
            Verify the applied term is the current mastership term to ensure the
          configuration has been synchronized following restarts.
          ∧ configuration.applied.term = mastership.term
          Verify the node's connection to the target.
          ∧ conn[n].connected
          ∧ mastership.conn = conn[n].id
          ∧ target.running
          Model successful and failed target update requests.
          ∧ ∨ ∧ LET values ≜ [p ∈ DOMAIN proposal[i].change.values ↦
                                  proposal[i].change.values[p] @@ [index ↦ i]]
                IN   ∧ target′ = [target EXCEPT !.values = values @@ target.values]
                     ∧ configuration′ = [configuration EXCEPT !.applied.index  = i,
                                                              !.applied.values = values @@
                                                                 configuration.applied.values]
                     ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Complete]
                     ∧ history′ = Append(history, [type ↦ Change, phase ↦ Apply, index ↦ i])
            ∨ ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Failed]
              ∧ UNCHANGED ⟨configuration, target, history⟩
      ∧ UNCHANGED ⟨mastership, conn⟩

CommitRollback(n, i) ≜
    ∧ ∨ ∧ proposal[i].rollback.commit = Pending
          ∧ i + 1 ∈ DOMAIN proposal ⇒ proposal[i + 1].rollback.commit = Complete
          ∧ ∨ ∧ proposal[i].change.commit = Pending
              ∧ proposal′ = [proposal EXCEPT ![i].change.commit   = Aborted,
                                             ![i].rollback.commit = Complete]
              ∧ UNCHANGED ⟨configuration⟩
            ∨ ∧ proposal[i].change.commit ≠ Pending
              ∧ configuration.committed.proposal = i
              ∧ configuration.committed.index = i
              ∧ configuration′ = [configuration EXCEPT !.committed.proposal = proposal[i].rollback.index]
              ∧ UNCHANGED ⟨proposal⟩
            ∨ ∧ proposal[i].change.commit ≠ Pending
              ∧ configuration.committed.proposal = proposal[i].rollback.index
              ∧ configuration.committed.index = i
              ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = InProgress]
              ∧ UNCHANGED ⟨configuration⟩
          ∧ UNCHANGED ⟨history⟩
      ∨ ∧ proposal[i].rollback.commit = InProgress
          ∧ ∨ ∧ configuration.committed.index ≠ configuration.committed.proposal
              ∧ LET index  ≜ proposal[i].rollback.index
                    values ≜ proposal[i].rollback.values @@ configuration.committed.values
                IN   ∧ configuration′ = [configuration EXCEPT !.committed.index  = index,
                                                              !.committed.values = values]
                     ∧ history′ = Append(history, [type ↦ Rollback, phase ↦ Commit, index ↦ i])
                     ∧ UNCHANGED ⟨proposal⟩
```

```
        ∨ ∧ configuration.committed.proposal = i
          ∧ configuration.committed.index = i
          ∧ proposal′ = [proposal EXCEPT ![i].rollback.commit = Complete]
          ∧ UNCHANGED ⟨configuration, history⟩
  ∧ UNCHANGED ⟨mastership, conn, target⟩

ApplyRollback(n, i) ≜
  ∧ ∨ ∧ proposal[i].rollback.apply = Pending
      ∧ proposal[i].rollback.commit = Complete
      ∧ ∀ j ∈ DOMAIN proposal : j > i ∧ proposal[j].phase ≠ None ⇒
          proposal[j].rollback.apply ∈ Finished
      ∧ ∨ ∧ proposal[i].change.apply = Pending
          ∧ proposal′ = [proposal EXCEPT ![i].change.apply   = Aborted,
                                         ![i].rollback.apply = Complete]
        ∨ ∧ proposal[i].change.apply ∈ Finished
          ∧ proposal′ = [proposal EXCEPT ![i].rollback.apply = InProgress]
      ∧ UNCHANGED ⟨configuration, target, history⟩
    ∨ ∧ proposal[i].rollback.apply = InProgress
       Verify the applied term is the current mastership term to ensure the
       configuration has been synchronized following restarts.
      ∧ configuration.applied.term = mastership.term
       Verify the node's connection to the target.
      ∧ conn[n].connected
      ∧ target.running
      ∧ target′ = [target EXCEPT !.values = proposal[i].rollback.values @@ target.values]
      ∧ LET index  ≜ proposal[i].rollback.index
            values ≜ proposal[i].rollback.values @@ configuration.applied.values
        IN
          ∧ configuration′ = [configuration EXCEPT !.applied.index  = index,
                                                   !.applied.values = values]
          ∧ proposal′ = [proposal EXCEPT ![i].rollback.apply = Complete]
          ∧ history′ = Append(history, [type ↦ Rollback, phase ↦ Apply, index ↦ i])
  ∧ UNCHANGED ⟨mastership, conn⟩

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

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

```
        $\mathit{status} \;\mapsto \mathit{Pending}]$
    $\land\ \mathit{mastership} = [\mathit{master} \mapsto \mathit{None},\ \mathit{term} \mapsto 0,\ \mathit{conn} \mapsto 0]$
    $\land\ \mathit{conn} = [n \in \mathit{Node} \mapsto [\mathit{id} \mapsto 0,\ \mathit{connected} \mapsto \text{FALSE}]]$
    $\land\ \mathit{target} = [$
        $\mathit{id} \qquad\;\; \mapsto 0,$
        $\mathit{values} \quad \mapsto [p \in \{\} \mapsto [\mathit{index} \mapsto 0,\ \mathit{value} \mapsto \mathit{None}]],$
        $\mathit{running} \mapsto \text{FALSE}]$
    $\land\ \mathit{history} = \langle\rangle$

$\mathit{Next} \;\triangleq$
    $\lor\ \exists\, i \in 1 \,..\, \mathit{NumProposals} :$
        $\lor\ \mathit{ProposeChange}(i)$
        $\lor\ \mathit{ProposeRollback}(i)$
    $\lor\ \exists\, n \in \mathit{Node},\ i \in \text{DOMAIN}\ \mathit{proposal} :$
        $\mathit{ProposalLog}\,!\,\mathit{Action}(\mathit{ReconcileProposal}(n,\ i),\ [\mathit{node} \mapsto n,\ \mathit{index} \mapsto i])$
    $\lor\ \exists\, n \in \mathit{Node} :$
        $\mathit{ConfigurationLog}\,!\,\mathit{Action}(\mathit{ReconcileConfiguration}(n),\ [\mathit{node} \mapsto n])$
    $\lor\ \exists\, n \in \mathit{Node} :$
        $\mathit{MastershipLog}\,!\,\mathit{Action}(\mathit{ReconcileMastership}(n),\ [\mathit{node} \mapsto n])$
    $\lor\ \exists\, n \in \mathit{Node} :$
       $\lor\ \mathit{ConnectNode}(n)$
       $\lor\ \mathit{DisconnectNode}(n)$
    $\lor\ \mathit{StartTarget}$
    $\lor\ \mathit{StopTarget}$

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

$\mathit{Mapping} \;\triangleq\; \text{INSTANCE}\ \mathit{Config}\ \text{WITH}$
   $\mathit{proposal} \leftarrow [i \in \text{DOMAIN}\ \mathit{proposal} \mapsto$
      $[\mathit{proposal}[i]\ \text{EXCEPT}\ !.\mathit{change}.\mathit{commit}$   $= \text{IF}\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{commit} = \mathit{InProgress}$
                                                    $\land\ \mathit{configuration}.\mathit{committed}.\mathit{index} = i$
                                                 $\text{THEN}\ \ \mathit{Complete}$
                                                 $\text{ELSE}\ \ \mathit{proposal}[i].\mathit{change}.\mathit{commit},$
                               $!.\mathit{change}.\mathit{apply}$      $= \text{IF}\ \land\ \mathit{proposal}[i].\mathit{change}.\mathit{apply} = \mathit{InProgress}$
                                                    $\land\ \mathit{configuration}.\mathit{applied}.\mathit{index} = i$
                                                 $\text{THEN}\ \ \mathit{Complete}$
                                                 $\text{ELSE}\ \ \mathit{proposal}[i].\mathit{change}.\mathit{apply},$
```

```
                                   !.rollback.commit  = IF  ∧ proposal[i].rollback.commit = InProgress
                                                            ∧ configuration.committed.index = proposal[i].rollback.ind
                                                          THEN  Complete
                                                          ELSE  proposal[i].rollback.commit,
                                   !.rollback.apply   = IF  ∧ proposal[i].rollback.apply = InProgress
                                                            ∧ configuration.applied.index = proposal[i].rollback.index
                                                          THEN  Complete
                                                          ELSE  proposal[i].rollback.apply]],
    configuration ← [
        committed ↦ [
            index   ↦ configuration.committed.index,
            values  ↦ configuration.committed.values],
        applied ↦ [
            index  ↦ configuration.applied.index,
            term   ↦ configuration.applied.term,
            target ↦ configuration.applied.target,
            values ↦ configuration.applied.values],
        status ↦ configuration.status]

Refinement ≜ Mapping!Spec

Order ≜ Mapping!Order

Consistency ≜ Mapping!Consistency

Liveness ≜ Mapping!Liveness

Sequential ≜ Mapping!Sequential
```