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

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

$Working \triangleq \{Pending, InProgress\}$

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

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 ≜ [ 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,
```

$values \mapsto Change$,
$running \mapsto$ BOOLEAN]

VARIABLE *proposal*

VARIABLE *configuration*

VARIABLE *mastership*

VARIABLE *conn*

VARIABLE *target*

VARIABLE *history*

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

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 $\mapsto [i \in \{i \in$ 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.

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

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 ∈ DOMAIN configuration.committed.values THEN
                                                   configuration.committed.values[p]
                                                ELSE
                                                   [index ↦ 0, value ↦ None]]
                        IN  proposal′ = [proposal EXCEPT ![i].rollback.index  = rollbackIndex,
                                                         ![i].rollback.values = rollbackValues,
                                                         ![i].change.commit = InProgress]
                          ∨ proposal′ = [proposal EXCEPT ![i].change.commit = Failed]
                       ∧ UNCHANGED ⟨configuration⟩
                  ∧ UNCHANGED ⟨history⟩
            ∨ ∧ proposal[i].change.commit = InProgress
              ∧ ∨ ∧ configuration.committed.index ≠ configuration.committed.proposal
                  ∧ 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.values = values]
                        ∧ history′ = Append(history, [type ↦ Change, phase ↦ Commit, index ↦ i])
                        ∧ UNCHANGED ⟨proposal⟩
                ∨ ∧ configuration.committed.proposal = i
                  ∧ configuration.committed.index = i
                  ∧ proposal′ = [proposal EXCEPT ![i].change.commit = Complete]
                  ∧ UNCHANGED ⟨configuration, history⟩
            ∨ ∧ proposal[i].change.commit = Failed
              ∧ configuration.committed.proposal = i
              ∧ configuration.committed.index ≠ i
              ∧ configuration′ = [configuration EXCEPT !.committed.index = i]
              ∧ UNCHANGED ⟨proposal, history⟩
        ∧ UNCHANGED ⟨target⟩

ApplyChange(n, i) ≜
    ∧ ∨ ∧ proposal[i].change.apply = Pending
        ∧ i − 1 ∈ DOMAIN proposal ⇒ proposal[i − 1].change.apply ∈ Finished
        ∧ proposal[i].rollback.apply = None
        ∧ ∨ ∧ proposal[i].change.commit = Complete
            ∧ configuration.applied.proposal < i
            ∧ configuration.applied.index = configuration.applied.proposal
            ∧ configuration′ = [configuration EXCEPT !.applied.proposal = i]
            ∧ UNCHANGED ⟨proposal⟩
          ∨ ∧ proposal[i].change.commit ∈ {Aborted, Failed}
            ∧ configuration.applied.proposal < i
            ∧ configuration.applied.index = configuration.applied.proposal
            ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Aborted]
            ∧ UNCHANGED ⟨configuration⟩
          ∨ ∧ configuration.applied.proposal = i
```

```
                ∧ configuration.applied.index ≠ i
                ∧ proposal′ = [proposal EXCEPT ![i].change.apply = InProgress]
                ∧ UNCHANGED ⟨configuration⟩
            ∧ UNCHANGED ⟨target, history⟩
        ∨ ∧ 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.
          ∧ ∨ ∧ configuration.applied.proposal = i
                ∧ configuration.applied.index ≠ i
                ∧ 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]
                        ∧ history′ = Append(history, [type ↦ Change, phase ↦ Apply, index ↦ i])
                        ∧ UNCHANGED ⟨proposal⟩
            ∨ ∧ configuration.applied.proposal = i
              ∧ configuration.applied.index ≠ i
              ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Failed]
              ∧ UNCHANGED ⟨configuration, target, history⟩
            ∨ ∧ configuration.applied.proposal = i
              ∧ configuration.applied.index = i
              ∧ proposal′ = [proposal EXCEPT ![i].change.apply = Complete]
              ∧ UNCHANGED ⟨configuration, target, history⟩
        ∨ ∧ proposal[i].change.apply = Failed
          ∧ configuration.applied.proposal = i
          ∧ configuration.applied.index ≠ i
          ∧ configuration′ = [configuration EXCEPT !.applied.index = i]
          ∧ UNCHANGED ⟨proposal, target, history⟩

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
```

$\land$ $configuration.committed.proposal = i$
$\land$ $configuration.committed.index = i$
$\land$ $configuration' = [configuration$ EXCEPT $!.committed.proposal = proposal[i].rollback.index]$
$\land$ UNCHANGED $\langle proposal \rangle$
$\lor$ $\land$ $proposal[i].change.commit \neq Pending$
$\land$ $configuration.committed.proposal = proposal[i].rollback.index$
$\land$ $configuration.committed.index = i$
$\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.commit = InProgress]$
$\land$ UNCHANGED $\langle configuration \rangle$
$\land$ UNCHANGED $\langle history \rangle$
$\lor$ $\land$ $proposal[i].rollback.commit = InProgress$
$\land$ $\lor$ $\land$ $configuration.committed.index \neq configuration.committed.proposal$
$\land$ LET $index \triangleq proposal[i].rollback.index$
$values \triangleq proposal[i].rollback.values \mathbin{@@} configuration.committed.values$
IN $\land$ $configuration' = [configuration$ EXCEPT $!.committed.index = index,$
$!.committed.values = values]$
$\land$ $history' = Append(history, [type \mapsto Rollback, phase \mapsto Commit, index \mapsto i])$
$\land$ UNCHANGED $\langle proposal \rangle$
$\lor$ $\land$ $configuration.committed.proposal = i$
$\land$ $configuration.committed.index = i$
$\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.commit = Complete]$
$\land$ UNCHANGED $\langle configuration, history \rangle$
$\land$ UNCHANGED $\langle target \rangle$

$ApplyRollback(n, i) \triangleq$
$\land$ $\lor$ $\land$ $proposal[i].rollback.apply = Pending$
$\land$ $i + 1 \in$ DOMAIN $proposal \Rightarrow proposal[i + 1].rollback.apply = Complete$
$\land$ $proposal[i].rollback.commit = Complete$
$\land$ $\lor$ $\land$ $proposal[i].change.apply = Pending$
$\land$ $proposal' = [proposal$ EXCEPT $![i].change.apply = Aborted,$
$![i].rollback.apply = Complete]$
$\land$ UNCHANGED $\langle configuration \rangle$
$\lor$ $\land$ $proposal[i].change.apply \neq Pending$
$\land$ $configuration.applied.proposal = i$
$\land$ $configuration.applied.index = i$
$\land$ $configuration' = [configuration$ EXCEPT $!.applied.proposal = proposal[i].rollback.index]$
$\land$ UNCHANGED $\langle proposal \rangle$
$\lor$ $\land$ $proposal[i].change.apply \neq Pending$
$\land$ $configuration.applied.proposal = proposal[i].rollback.index$
$\land$ $configuration.applied.index = i$
$\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.apply = InProgress]$
$\land$ UNCHANGED $\langle configuration \rangle$
$\land$ UNCHANGED $\langle target, history \rangle$
$\lor$ $\land$ $proposal[i].rollback.apply = InProgress$
$\land$ $\lor$ $\land$ $configuration.applied.proposal = proposal[i].rollback.index$

$\land$ $configuration.applied.index = i$

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.

$\land$ $conn[n].connected$

$\land$ $target.running$

$\land$ $target' = [target \text{ EXCEPT } !.values = proposal[i].rollback.values \mathbin{@@} target.values]$

$\land$ $configuration' = [configuration \text{ EXCEPT } !.applied.index = proposal[i].rollback.index,$
$!.applied.values = proposal[i].rollback.values \mathbin{@@} configuration.applied.values]$

$\land$ $history' = Append(history, [type \mapsto Rollback, phase \mapsto Apply, index \mapsto i])$

$\land$ UNCHANGED $\langle proposal \rangle$

$\lor$ $\land$ $configuration.applied.proposal = proposal[i].rollback.index$

$\land$ $configuration.applied.index = proposal[i].rollback.index$

$\land$ $proposal' = [proposal \text{ EXCEPT } ![i].rollback.apply = Complete]$

$\land$ UNCHANGED $\langle configuration, target, history \rangle$

$ReconcileProposal(n, i) \triangleq$

$\land$ $mastership.master = n$

$\land$ $\lor$ $CommitChange(n, i)$

$\lor$ $ApplyChange(n, i)$

$\lor$ $CommitRollback(n, i)$

$\lor$ $ApplyRollback(n, i)$

$\land$ UNCHANGED $\langle mastership, conn \rangle$

This section models changes to the proposal queue.

Propose change at index 'i'

$ProposeChange(i) \triangleq$

$\land$ $proposal[i].phase = None$

$\land$ $i - 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].phase \neq None$

$\land$ $\exists\, p \in Path, v \in AllValues :$

$\land$ $proposal' = [proposal \text{ EXCEPT } ![i].phase = Change,$
$![i].change.values = (p :> [value \mapsto v]),$
$![i].change.commit = Pending,$
$![i].change.apply = Pending]$

$\land$ UNCHANGED $\langle configuration, mastership, conn, target, history \rangle$

Rollback proposed change at index 'i'

$ProposeRollback(i) \triangleq$

$\land$ $proposal[i].phase = Change$

$\land$ $proposal' = [proposal \text{ EXCEPT } ![i].phase = Rollback,$
$![i].rollback.commit = Pending,$
$![i].rollback.apply = Pending]$

$\land$ UNCHANGED $\langle configuration,\ mastership,\ conn,\ target,\ history\rangle$

Formal specification, constraints, and theorems.

$Init \triangleq$

$\land\ proposal = [$
$\quad i \in 1\,..\,NumProposals \mapsto [$
$\quad\quad phase \quad \mapsto None,$
$\quad\quad change \quad \mapsto [$
$\quad\quad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
$\quad\quad\quad commit \mapsto None,$
$\quad\quad\quad apply \quad \mapsto None],$
$\quad\quad rollback \mapsto [$
$\quad\quad\quad index \quad \mapsto 0,$
$\quad\quad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
$\quad\quad\quad commit \mapsto None,$
$\quad\quad\quad apply \quad \mapsto None]]]$
$\land\ configuration = [$
$\quad committed \mapsto [$
$\quad\quad proposal \mapsto 0,$
$\quad\quad index \quad \mapsto 0,$
$\quad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
$\quad applied \mapsto [$
$\quad\quad proposal \mapsto 0,$
$\quad\quad index \quad \mapsto 0,$
$\quad\quad term \quad \mapsto 0,$
$\quad\quad target \quad \mapsto 0,$
$\quad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]]],$
$\quad status \ \mapsto Pending]$
$\land\ mastership = [master \mapsto None,\ term \mapsto 0,\ conn \mapsto 0]$
$\land\ conn = [n \in Node \mapsto [id \mapsto 0,\ connected \mapsto \text{FALSE}]]$
$\land\ target = [$
$\quad id \quad\quad \mapsto 0,$
$\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto None]],$
$\quad running \mapsto \text{FALSE}]$
$\land\ history = \langle\rangle$

$Next \triangleq$

$\lor \exists\, i \in 1\,..\,NumProposals :$
$\quad \lor\ ProposeChange(i)$
$\quad \lor\ ProposeRollback(i)$
$\lor \exists\, n \in Node,\ i \in \text{DOMAIN}\ proposal :$
$\quad ProposalLog!Action(ReconcileProposal(n,\ i),\ [node \mapsto n,\ index \mapsto i])$
$\lor \exists\, n \in Node :$
$\quad ConfigurationLog!Action(ReconcileConfiguration(n),\ [node \mapsto n])$

$\lor \exists\, n \in Node :$
$\quad MastershipLog!Action(ReconcileMastership(n),\ [node \mapsto n])$
$\lor \exists\, n \in Node :$
$\quad \lor\ ConnectNode(n)$
$\quad \lor\ DisconnectNode(n)$
$\lor\ StartTarget$
$\lor\ StopTarget$

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

$Mapping\ \triangleq$ INSTANCE $Config$ WITH
$\quad proposal \leftarrow [i \in$ DOMAIN $proposal \mapsto [$
$\quad\quad phase \quad \mapsto proposal[i].phase,$
$\quad\quad values \quad \mapsto [p \in$ DOMAIN $proposal[i].change.values \mapsto proposal[i].change.values[p].value],$
$\quad\quad change \quad \mapsto [$
$\quad\quad\quad commit \mapsto$ IF $\land\ proposal[i].change.commit = InProgress$
$\quad\quad\quad\quad\quad\quad\quad\quad \land\ configuration.committed.index = i$
$\quad\quad\quad\quad\quad\quad$ THEN $Complete$
$\quad\quad\quad\quad\quad\quad$ ELSE $proposal[i].change.commit,$
$\quad\quad\quad apply \quad \mapsto$ IF $\land\ proposal[i].change.apply = InProgress$
$\quad\quad\quad\quad\quad\quad\quad\quad \land\ configuration.applied.index = i$
$\quad\quad\quad\quad\quad\quad$ THEN $Complete$
$\quad\quad\quad\quad\quad\quad$ ELSE $proposal[i].change.apply],$
$\quad\quad rollback \mapsto [$
$\quad\quad\quad commit \mapsto$ IF $\land\ proposal[i].rollback.commit = InProgress$
$\quad\quad\quad\quad\quad\quad\quad\quad \land\ configuration.committed.index = proposal[i].rollback.index$
$\quad\quad\quad\quad\quad\quad$ THEN $Complete$
$\quad\quad\quad\quad\quad\quad$ ELSE $proposal[i].rollback.commit,$
$\quad\quad\quad apply \quad \mapsto$ IF $\land\ proposal[i].rollback.apply = InProgress$
$\quad\quad\quad\quad\quad\quad\quad\quad \land\ configuration.applied.index = proposal[i].rollback.index$
$\quad\quad\quad\quad\quad\quad$ THEN $Complete$
$\quad\quad\quad\quad\quad\quad$ ELSE $proposal[i].rollback.apply]],$
$\quad configuration \leftarrow [$
$\quad\quad committed \mapsto [$
$\quad\quad\quad values \quad \mapsto configuration.committed.values],$
$\quad\quad applied \mapsto [$

$$
\begin{array}{l}
\quad term \;\mapsto configuration.applied.term, \\
\quad target \;\mapsto configuration.applied.target, \\
\quad values \;\mapsto configuration.applied.values], \\
status \mapsto configuration.status]
\end{array}
$$

$Refinement \triangleq Mapping!Spec$

$Order \triangleq Mapping!Order$

$Consistency \triangleq Mapping!Consistency$

$Liveness \triangleq Mapping!Liveness$

$Sequential \triangleq Mapping!Sequential$