MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

This section specifies constant parameters for the model.

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

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

$Working \triangleq \{Pending, InProgress\}$
$Finished \triangleq \{Complete, Aborted, Failed\}$

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

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.

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

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

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

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

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

VARIABLE $proposal$

VARIABLE *configuration*

VARIABLE *mastership*

VARIABLE *conn*

VARIABLE *target*

VARIABLE *history*

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

This section models configuration target.

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

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

This section models nodes connection to the configuration target.

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

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

This section models *mastership* reconciliation.

$ReconcileMastership(n) \triangleq$
$\quad \land \lor \land conn[n].connected$
$\qquad\quad \land mastership.master = None$

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

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.

$CommitChange(n,\ i) \triangleq$
$\land\ \lor\ \land\ proposal[i].change.commit = Pending$
$\land\ proposal[i].rollback.commit = None$
$\land\ \forall\, j \in \text{DOMAIN } proposal : j < i \Rightarrow$
$\land\ proposal[j].change.commit \in Finished$
$\land\ proposal[j].rollback.commit \neq InProgress$
$\land\ proposal' = [proposal \text{ EXCEPT } ![i].change.commit = InProgress]$
$\land \text{ UNCHANGED } \langle configuration,\ history\rangle$
$\lor\ \land\ proposal[i].change.commit = InProgress$

Changes are validated during the commit phase. If a change fails validation,
it will be marked failed before being applied to the configuration.
If all the change values are valid, record the changes required to roll
back the proposal and the index to which the rollback changes
will roll back the configuration.

$\land\ \lor\ \land \text{ LET } values \triangleq [p \in \text{DOMAIN } proposal[i].values \mapsto$
$[index \mapsto i,\ value \mapsto proposal[i].values[p]]]\ @@$
$configuration.committed.values$
$\text{IN} \quad \land\ configuration' = [configuration \text{ EXCEPT } !.committed.values = values]$

$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.commit = Complete]$
$\land\ history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])$
$\lor\ \land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.commit = Failed]$
$\land\ \text{UNCHANGED}\ \langle configuration, history\rangle$
$\land\ \text{UNCHANGED}\ \langle mastership, conn, target\rangle$

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

$CommitRollback(n, i) \triangleq$
$\land\ \lor\ \land\ proposal[i].rollback.commit = Pending$
$\land\ \forall\, j \in \text{DOMAIN}\ proposal : j > i \land proposal[j].phase \neq None \Rightarrow$
$proposal[j].rollback.commit = Complete$
$\land\ \lor\ \land\ proposal[i].change.commit = Pending$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.commit \quad = Aborted,$

```
                                                    ![i].rollback.commit  = Complete]
            ∨ ∧ proposal[i].change.commit ∈ Finished
              ∧ proposal' = [proposal EXCEPT ![i].rollback.commit = InProgress]
          ∧ UNCHANGED ⟨configuration, history⟩
        ∨ ∧ proposal[i].rollback.commit = InProgress
          ∧ LET changes ≜ {j ∈ DOMAIN proposal :
                             ∧ j < i
                             ∧ proposal[j].change.commit = Complete
                             ∧ proposal[j].rollback.commit ≠ Complete}
                paths   ≜ {p ∈ DOMAIN configuration.committed.values :
                             ∃ j ∈ changes : p ∈ DOMAIN proposal[j].values}
                indexes ≜ [p ∈ paths ↦ CHOOSE j ∈ changes :
                             ∧ p ∈ DOMAIN proposal[j].values
                             ∧ ¬∃ k ∈ changes : k > j ∧ p ∈ DOMAIN proposal[k].values]
                values  ≜ [p ∈ DOMAIN configuration.committed.values ↦
                             IF p ∈ paths THEN
                               [index ↦ indexes[p], value ↦ proposal[indexes[p]].values[p]]
                             ELSE
                               [index ↦ 0, value ↦ None]]
            IN
              ∧ configuration' = [configuration EXCEPT !.committed.values = values]
              ∧ proposal' = [proposal EXCEPT ![i].rollback.commit = Complete]
              ∧ history' = Append(history, [type ↦ Rollback, phase ↦ Commit, index ↦ i])
    ∧ 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
        ∧ LET changes ≜ {j ∈ DOMAIN proposal :
                           ∧ j < i
```

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad \wedge\ proposal[j].change.apply = Complete \\
\qquad\qquad\qquad\qquad \wedge\ proposal[j].rollback.apply \neq Complete\} \\
\qquad\quad paths \triangleq \{p \in \text{DOMAIN } configuration.applied.values : \\
\qquad\qquad\qquad\qquad \exists\, j \in changes : p \in \text{DOMAIN } proposal[j].values\} \\
\qquad\quad indexes \triangleq [p \in paths \mapsto \text{CHOOSE } j \in changes : \\
\qquad\qquad\qquad\qquad \wedge\ p \in \text{DOMAIN } proposal[j].values \\
\qquad\qquad\qquad\qquad \wedge\ \neg\exists\, k \in changes : k > j \wedge p \in \text{DOMAIN } proposal[k].values] \\
\qquad\quad values \triangleq [p \in \text{DOMAIN } configuration.applied.values \mapsto \\
\qquad\qquad\qquad\qquad \text{IF } p \in paths \text{ THEN} \\
\qquad\qquad\qquad\qquad\quad [index \mapsto indexes[p],\ value \mapsto proposal[indexes[p]].values[p]] \\
\qquad\qquad\qquad\qquad \text{ELSE} \\
\qquad\qquad\qquad\qquad\quad [index \mapsto 0,\ value \mapsto None]] \\
\text{IN} \\
\qquad \wedge\ target' = [target \text{ EXCEPT } !.values = values] \\
\qquad \wedge\ configuration' = [configuration \text{ EXCEPT } !.applied.values = values] \\
\qquad \wedge\ proposal' = [proposal \text{ EXCEPT } ![i].rollback.apply = Complete] \\
\qquad \wedge\ history' = Append(history, [type \mapsto Rollback,\ phase \mapsto Apply,\ index \mapsto i]) \\
\wedge \text{ UNCHANGED } \langle mastership,\ conn\rangle
\end{array}
$$

$$
\begin{array}{l}
ReconcileProposal(n,\ i) \triangleq \\
\quad \wedge\ mastership.master = n \\
\quad \wedge\ \vee\ CommitChange(n,\ i) \\
\qquad \vee\ ApplyChange(n,\ i) \\
\qquad \vee\ CommitRollback(n,\ i) \\
\qquad \vee\ ApplyRollback(n,\ i) \\
\quad \wedge \text{ UNCHANGED } \langle mastership,\ conn\rangle
\end{array}
$$

This section models changes to the proposal queue.

Propose change at index 'i'

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

Rollback proposed change at index 'i'

$$
\begin{array}{l}
ProposeRollback(i) \triangleq \\
\quad \wedge\ proposal[i].phase = Change \\
\quad \wedge\ proposal' = [proposal \text{ EXCEPT } ![i].phase \quad\quad\quad = Rollback, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad ![i].rollback.commit = Pending,
\end{array}
$$

$$!\,[i].rollback.apply \quad = Pending]$$
$$\land \text{ UNCHANGED } \langle configuration,\ mastership,\ conn,\ target,\ history \rangle$$

Formal specification, constraints, and theorems.

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

$$Next \triangleq$$
$$\lor \exists\, i \in 1\,..\,NumProposals :$$
$$\quad \lor ProposeChange(i)$$
$$\quad \lor ProposeRollback(i)$$
$$\lor \exists\, n \in Node,\ i \in \text{DOMAIN } proposal : ReconcileProposal(n,\ i)$$
$$\lor \exists\, n \in Node : ReconcileConfiguration(n)$$
$$\lor \exists\, n \in Node : ReconcileMastership(n)$$
$$\lor \exists\, n \in Node :$$
$$\quad \lor ConnectNode(n)$$
$$\quad \lor DisconnectNode(n)$$
$$\lor StartTarget$$
$$\lor StopTarget$$

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

$$
\begin{array}{l}
IsOrderedChange(p,\, i) \triangleq \\
\quad \land history[i].type = Change \\
\quad \land history[i].phase = p \\
\quad \land \neg\exists\, j \in \text{DOMAIN } history : \\
\qquad \land j < i \\
\qquad \land history[j].type = Change \\
\qquad \land history[j].phase = p \\
\qquad \land history[j].index \geq history[i].index
\end{array}
$$

$$
\begin{array}{l}
IsOrderedRollback(p,\, i) \triangleq \\
\quad \land history[i].type = Rollback \\
\quad \land history[i].phase = p \\
\quad \land \neg\exists\, j \in \text{DOMAIN } history : \\
\qquad \land j < i \\
\qquad \land history[j].type = Change \\
\qquad \land history[j].phase = p \\
\qquad \land history[j].index > history[i].index \\
\qquad \land \neg\exists\, k \in \text{DOMAIN } history : \\
\qquad\qquad \land k > j \\
\qquad\qquad \land k < i \\
\qquad\qquad \land history[k].type = Rollback \\
\qquad\qquad \land history[k].phase = p \\
\qquad\qquad \land history[k].index = history[j].index
\end{array}
$$

$$
\begin{array}{l}
Order \triangleq \\
\quad \land\ \forall\, i \in \text{DOMAIN } history : \\
\qquad \lor IsOrderedChange(Commit,\, i) \\
\qquad \lor IsOrderedChange(Apply,\, i) \\
\qquad \lor IsOrderedRollback(Commit,\, i) \\
\qquad \lor IsOrderedRollback(Apply,\, i) \\
\quad \land\ \forall\, i \in \text{DOMAIN } proposal : \\
\qquad \land proposal[i].change.apply = Failed \\
\qquad \land proposal[i].rollback.apply \neq Complete \\
\qquad \Rightarrow \forall\, j \in \text{DOMAIN } proposal : j > i \Rightarrow \\
\qquad\qquad proposal[j].change.apply \in \{None,\, Pending,\, Aborted\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{AdditiveChanges} \triangleq \\
\quad \land \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{commit} = \mathit{Pending} \\
\qquad \land\ \mathit{proposal}'[i].\mathit{change}.\mathit{commit} = \mathit{InProgress} \\
\qquad \Rightarrow \forall\, j \in \text{DOMAIN } \mathit{proposal} : j < i \ \Rightarrow \\
\qquad\quad \land\ \mathit{proposal}[j].\mathit{change}.\mathit{commit} \in \mathit{Finished} \\
\qquad\quad \land\ \mathit{proposal}[j].\mathit{rollback}.\mathit{commit} \neq \mathit{InProgress} \\
\quad \land \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{apply} = \mathit{Pending} \\
\qquad \land\ \mathit{proposal}'[i].\mathit{change}.\mathit{apply} = \mathit{InProgress} \\
\qquad \Rightarrow \forall\, j \in \text{DOMAIN } \mathit{proposal} : j < i \ \Rightarrow \\
\qquad\quad \land\ \mathit{proposal}[j].\mathit{change}.\mathit{apply} \in \mathit{Finished} \\
\qquad\quad \land\ \mathit{proposal}[j].\mathit{rollback}.\mathit{apply} \neq \mathit{InProgress}
\end{array}
$$

$$
\begin{array}{l}
\mathit{SubtractiveRollbacks} \triangleq \\
\quad \land \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{commit} = \mathit{Pending} \\
\qquad \land\ \mathit{proposal}'[i].\mathit{rollback}.\mathit{commit} = \mathit{InProgress} \\
\qquad \Rightarrow \forall\, j \in \text{DOMAIN } \mathit{proposal} : j\ > i \land \mathit{proposal}[j].\mathit{phase} \neq \mathit{None} \Rightarrow \\
\qquad\quad \mathit{proposal}[j].\mathit{rollback}.\mathit{commit} = \mathit{Complete} \\
\quad \land \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{apply} = \mathit{Pending} \\
\qquad \land\ \mathit{proposal}'[i].\mathit{rollback}.\mathit{apply} = \mathit{InProgress} \\
\qquad \Rightarrow \forall\, j \in \text{DOMAIN } \mathit{proposal} : j > i \land \mathit{proposal}[j].\mathit{phase} \neq \mathit{None} \Rightarrow \\
\qquad\quad \mathit{proposal}[j].\mathit{rollback}.\mathit{apply} = \mathit{Complete}
\end{array}
$$

$$
\mathit{Sequential} \triangleq \Box[\mathit{AdditiveChanges} \land \mathit{SubtractiveRollbacks}]_{\langle \mathit{proposal} \rangle}
$$

$$
\begin{array}{l}
\mathit{Consistency} \triangleq \\
\quad \land\ \mathit{target}.\mathit{running} \\
\quad \land\ \mathit{configuration}.\mathit{status} = \mathit{Complete} \\
\quad \land\ \mathit{configuration}.\mathit{applied}.\mathit{target} = \mathit{target}.\mathit{id} \\
\quad \Rightarrow \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{apply} = \mathit{Complete} \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{apply} \neq \mathit{Complete} \\
\qquad \Rightarrow \forall\, p \in \text{DOMAIN } \mathit{proposal}[i].\mathit{values} : \\
\qquad\quad \land \neg \exists\, j \in \text{DOMAIN } \mathit{proposal} : \\
\qquad\qquad \land\ j > i \\
\qquad\qquad \land\ \mathit{proposal}[j].\mathit{change}.\mathit{apply} = \mathit{Complete} \\
\qquad\qquad \land\ \mathit{proposal}[j].\mathit{rollback}.\mathit{apply} \neq \mathit{Complete} \\
\qquad\quad \Rightarrow\ \land\ p \in \text{DOMAIN } \mathit{target}.\mathit{values} \\
\qquad\qquad\ \land\ \mathit{target}.\mathit{values}[p].\mathit{value} = \mathit{proposal}[i].\mathit{values}[p] \\
\qquad\qquad\ \land\ \mathit{target}.\mathit{values}[p].\mathit{index} = i
\end{array}
$$

$$
\mathit{Safety} \triangleq \Box(\mathit{Order} \land \mathit{Consistency})
$$

THEOREM $Spec \Rightarrow Safety$

$$
\begin{array}{l}
Termination \triangleq \\
\quad \forall\, i \in 1 \,..\, NumProposals : \\
\qquad \land\, proposal[i].change.commit = Pending \leadsto \\
\qquad\qquad proposal[i].change.commit \in Finished \\
\qquad \land\, proposal[i].change.apply = Pending \leadsto \\
\qquad\qquad proposal[i].change.apply \in Finished \\
\qquad \land\, proposal[i].rollback.commit = Pending \leadsto \\
\qquad\qquad proposal[i].rollback.commit \in Finished \\
\qquad \land\, proposal[i].rollback.apply = Pending \leadsto \\
\qquad\qquad proposal[i].rollback.apply \in Finished
\end{array}
$$

$$Liveness \triangleq Termination$$

THEOREM $Spec \Rightarrow Liveness$