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

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

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

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

CONSTANT $NumProposals$

ASSUME $NumProposals \in Nat$

This section defines model state variables.

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

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

$mastership \triangleq [$
$\quad master \mapsto$ 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 $history$

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

This section models configuration target.

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

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

This section models nodes connection to the configuration target.

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

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

This section models *mastership* reconciliation.

$ReconcileMastership(n) \triangleq$
$\land \lor \land conn[n].connected$
$\qquad \land mastership.master = None$
$\qquad \land mastership' = [master \mapsto n,\ term \mapsto mastership.term + 1,\ conn \mapsto conn[n].id]$
$\quad \lor \land \neg conn[n].connected$
$\qquad \land mastership.master = n$

$\land\ mastership' = [mastership \text{ EXCEPT } !.master = None]$
$\land \text{ UNCHANGED } \langle proposal,\ configuration,\ conn,\ target,\ history\rangle$

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.

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

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.

$\land\ \lor \text{ LET } rollbackIndex \triangleq configuration.committed.index$
$rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
$\text{IF } p \in \text{DOMAIN } configuration.committed.values \text{ THEN}$

```
                                        configuration.committed.values[p]
                                    ELSE
                                        [index ↦ 0, value ↦ None]]
            changeValues  ≜  [p ∈ DOMAIN proposal[i].change.values ↦
                                proposal[i].change.values[p] @@ [index ↦ i]]
        IN  ∧ configuration' = [configuration EXCEPT !.committed.index  = i,
                                                     !.committed.values = changeValues]
            ∧ proposal' = [proposal EXCEPT ![i].change.values    = changeValues,
                                           ![i].rollback.values  = rollbackValues,
                                           ![i].change.phase     = Apply,
                                           ![i].change.state     = Pending]
            ∧ history' = Append(history, [type ↦ Change, phase ↦ Commit, index ↦ i])
      ∨ ∧ proposal' = [proposal EXCEPT ![i].change.state = Failed]
        ∧ UNCHANGED ⟨configuration, history⟩
  ∧ UNCHANGED ⟨mastership, conn, target⟩

ApplyChange(n, i) ≜
  ∧ proposal[i].change.phase = Apply
  ∧ ∨ ∧ proposal[i].change.state = Pending
      ∧ proposal[i].rollback.phase = None
      ∧ ∀ j ∈ DOMAIN proposal : j < i ⇒
          ∨ ∧ proposal[j].change.phase = Commit
            ∧ proposal[j].rollback.state = Complete
          ∨ ∧ proposal[j].change.phase = Apply
            ∧ proposal[j].change.state  = Complete
            ∧ proposal[j].rollback.phase = None
          ∨ ∧ proposal[j].change.phase = Apply
            ∧ proposal[j].change.state  = Failed
            ∧ proposal[j].rollback.phase = Apply
            ∧ proposal[j].rollback.state  = Complete
      ∧ proposal' = [proposal EXCEPT ![i].change.state = InProgress]
      ∧ UNCHANGED ⟨configuration, target, history⟩
    ∨ ∧ proposal[i].change.state = 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.
      ∧ ∨ ∧ target' = [target EXCEPT !.values = proposal[i].change.values @@ target.values]
            ∧ configuration' = [configuration EXCEPT !.applied.index  = i,
                                                     !.applied.values = proposal[i].change.values @@
                                                                        configuration.applied.values]
```

$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.state = Complete]$
$\land\ history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$

If the proposal could not be applied, mark it failed but do not update the last applied index. The proposal must be rolled back before new proposals can be applied to the configuration/target.

$\lor\ \land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.state = Failed]$
$\land\ \text{UNCHANGED}\ \langle configuration, target, history\rangle$

$\land\ \text{UNCHANGED}\ \langle mastership, conn\rangle$

$CommitRollback(n, i)\ \triangleq$

$\land\ proposal[i].rollback.phase = Commit$
$\land\ \lor\ \land\ proposal[i].rollback.state = Pending$
$\land\ \forall\, j \in \text{DOMAIN}\ proposal : j > i \Rightarrow$
$\lor\ \land\ proposal[j].rollback.phase = Commit$
$\land\ proposal[j].rollback.state \in Done$
$\lor\ proposal[j].rollback.phase = Apply$
$\land\ \lor\ \land\ proposal[i].change.phase = Commit$
$\land\ proposal[i].change.state = Pending$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.state = Aborted,$
$![i].rollback.state = Complete]$
$\lor\ \land\ proposal[i].change.phase = Commit$
$\land\ proposal[i].change.state \in Done$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].rollback.state = Complete]$
$\lor\ \land\ proposal[i].change.phase = Apply$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].rollback.state = InProgress]$
$\land\ \text{UNCHANGED}\ \langle configuration, history\rangle$
$\lor\ \land\ proposal[i].rollback.state = InProgress$
$\land\ \text{LET}\ index \triangleq proposal[i].rollback.index$
$values \triangleq proposal[i].rollback.values\ @@\ configuration.committed.values$
$\text{IN}\ \land\ configuration' = [configuration\ \text{EXCEPT}\ !.committed.index = index,$
$!.committed.values = values]$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].rollback.phase = Apply,$
$![i].rollback.state = Pending]$
$\land\ history' = Append(history, [type \mapsto Rollback, phase \mapsto Commit, index \mapsto i])$
$\land\ \text{UNCHANGED}\ \langle mastership, conn, target\rangle$

$ApplyRollback(n, i)\ \triangleq$

$\land\ proposal[i].rollback.phase = Apply$
$\land\ \lor\ \land\ proposal[i].rollback.state = Pending$
$\land\ \forall\, j \in \text{DOMAIN}\ proposal : j > i \Rightarrow$
$\land\ proposal[j].rollback.phase = Apply$
$\land\ proposal[j].rollback.state \in Done$
$\land\ \lor\ \land\ proposal[i].change.phase = Apply$
$\land\ proposal[i].change.state = Pending$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].change.state = Aborted,$

$!\,[i].rollback.state \;\; = Complete]$

$\lor \;\land\; proposal[i].change.phase = Apply$

$\land\; proposal[i].change.state \;\in Done$

$\land\; proposal' = [proposal \text{ EXCEPT } !\,[i].rollback.state = InProgress]$

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

$\lor \;\land\; proposal[i].rollback.state = InProgress$

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.

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

$\land\; target.running$

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

$\land$ LET $index \triangleq proposal[i].rollback.index$

$\quad values \triangleq proposal[i].rollback.values \;@@\; configuration.applied.values$

IN

$\land\; configuration' = [configuration \text{ EXCEPT } !.applied.index \;\; = index,$

$\qquad !.applied.values = values]$

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

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

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

$ReconcileProposal(n, i) \triangleq$

$\land\; mastership.master = n$

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

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

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

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

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

This section models changes to the proposal queue.

Propose change at index 'i'

$ProposeChange(i) \triangleq$

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

$\land\; i - 1 \in$ DOMAIN $proposal \Rightarrow proposal[i-1].change.phase \neq None$

$\land\; \exists\, p \in Path, v \in Value \cup \{None\}:$

$\land\; proposal' = [proposal \text{ EXCEPT } !\,[i].change.phase = Commit,$

$\qquad !\,[i].change.state \;\; = Pending,$

$\qquad !\,[i].change.values = (p :> [value \mapsto v])]$

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

Rollback proposed change at index 'i'

$ProposeRollback(i) \triangleq$

$\land\; proposal[i].change.phase \neq None$

$\land$ $proposal[i].rollback.phase = None$
$\land$ $proposal' = [proposal$ EXCEPT $![i].rollback.phase = Commit,$
$![i].rollback.state = Pending]$
$\land$ UNCHANGED $\langle configuration, mastership, conn, target, history\rangle$

Formal specification, constraints, and theorems.

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

$Next \triangleq$
$\lor \exists\, i \in 1 \,..\, NumProposals :$
$\lor ProposeChange(i)$
$\lor ProposeRollback(i)$
$\lor \exists\, n \in Node,\ i \in$ DOMAIN $proposal : ReconcileProposal(n,\ i)$
$\lor \exists\, n \in Node : ReconcileConfiguration(n)$
$\lor \exists\, n \in Node : ReconcileMastership(n)$
$\lor \exists\, n \in Node :$

$$
\begin{array}{l}
\quad \lor \mathit{ConnectNode}(n) \\
\quad \lor \mathit{DisconnectNode}(n) \\
\lor \mathit{StartTarget} \\
\lor \mathit{StopTarget}
\end{array}
$$

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

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

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

$$
\begin{array}{l}
\mathit{Order} \triangleq \\
\quad \land\ \forall\, i \in \textsc{domain}\ \mathit{history} : \\
\qquad \lor \mathit{IsOrderedChange}(\mathit{Commit},\ i) \\
\qquad \lor \mathit{IsOrderedChange}(\mathit{Apply},\ i) \\
\qquad \lor \mathit{IsOrderedRollback}(\mathit{Commit},\ i) \\
\qquad \lor \mathit{IsOrderedRollback}(\mathit{Apply},\ i)
\end{array}
$$

$$
\begin{array}{l}
\land\ \forall\, i \in \text{DOMAIN } proposal : \\
\quad \land\ proposal[i].change.phase = Apply \\
\quad \land\ proposal[i].change.state\ \notin \{Pending,\ Aborted\} \\
\quad \Rightarrow \forall\, j \in \text{DOMAIN } proposal : \\
\qquad \land\ j < i \\
\qquad \land\ proposal[j].change.phase = Apply \\
\qquad \land\ proposal[j].change.state\ = Failed \\
\qquad \Rightarrow\ \land\ proposal[j].rollback.phase = Apply \\
\qquad\quad \land\ proposal[j].rollback.state\ = Complete
\end{array}
$$

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

$$Safety\ \triangleq\ \Box(Order \land Consistency)$$

THEOREM $Spec \Rightarrow Safety$

$$
\begin{array}{l}
Termination\ \triangleq \\
\quad \forall\, i \in 1\,..\,NumProposals : \\
\qquad \land\ proposal[i].change.phase = Commit \rightsquigarrow \\
\qquad\quad \lor\ \land\ proposal[i].change.phase = Commit \\
\qquad\qquad \land\ proposal[i].change.state\ \in Done \\
\qquad\quad \lor\ proposal[i].change.phase = Apply \\
\qquad \land\ proposal[i].change.phase = Apply \rightsquigarrow \\
\qquad\quad \land\ proposal[i].change.state \in Done \\
\qquad \land\ proposal[i].rollback.phase = Commit \rightsquigarrow \\
\qquad\quad \lor\ \land\ proposal[i].rollback.phase = Commit \\
\qquad\qquad \land\ proposal[i].rollback.state\ \in Done \\
\qquad\quad \lor\ proposal[i].rollback.phase = Apply \\
\qquad \land\ proposal[i].rollback.phase = Apply \rightsquigarrow \\
\qquad\quad \land\ proposal[i].rollback.state \in Done
\end{array}
$$

$$\land \lor \land proposal[i].change.phase = Commit$$
$$\quad \land proposal[i].change.state \in Done$$
$$\quad \lor proposal[i].change.phase = Apply$$

$Liveness \triangleq Termination$

THEOREM $Spec \Rightarrow Liveness$

* Modification History
* Last modified *Fri Apr* 21 18:30:03 *PDT* 2023 by *jhalterm*
* Last modified *Mon Feb* 21 01:32:07 *PST* 2022 by *jordanhalterman*
* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*