MODULE *Proposal*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant
CONSTANT *Nil*

Phase constants
CONSTANTS
Commit,
Apply

Phase $\triangleq$
{*Nil*,
Commit,
Apply}

Status constants
CONSTANTS
Pending,
InProgress,
Complete,
Failed

Status $\triangleq$
{*Nil*,
Pending,
InProgress,
Complete,
Failed}

The set of all nodes
CONSTANT *Node*

Variables defined by other modules.
VARIABLES
configuration,
mastership,
target

A record of per-target proposals
VARIABLE *proposal*

$$
\begin{array}{l}
\mathit{TypeOK} \triangleq \\
\quad \forall\, i \in \text{DOMAIN } \mathit{proposal} : \\
\qquad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{phase} \in \mathit{Phase} \\
\qquad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{state}\ \in \mathit{Status} \\
\qquad \land\ \forall\, p \in \text{DOMAIN } \mathit{proposal}[i].\mathit{change}.\mathit{values} : \\
\qquad\quad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{values}[p].\mathit{index} \in \mathit{Nat} \\
\qquad\quad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{values}[p].\mathit{value} \neq \mathit{Nil} \Rightarrow \\
\qquad\qquad \mathit{proposal}[i].\mathit{change}.\mathit{values}[p].\mathit{value} \in \text{STRING} \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{phase} \in \mathit{Phase} \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{state}\ \in \mathit{Status} \\
\qquad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{revision} \in \mathit{Nat} \\
\qquad \land\ \forall\, p \in \text{DOMAIN } \mathit{proposal}[i].\mathit{rollback}.\mathit{values} : \\
\qquad\quad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{values}[p].\mathit{index} \in \mathit{Nat} \\
\qquad\quad \land\ \mathit{proposal}[i].\mathit{rollback}.\mathit{values}[p].\mathit{value} \neq \mathit{Nil} \Rightarrow \\
\qquad\qquad \mathit{proposal}[i].\mathit{rollback}.\mathit{values}[p].\mathit{value} \in \text{STRING}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Test} \triangleq \text{INSTANCE } \mathit{Test} \text{ WITH} \\
\quad \mathit{File} \leftarrow \text{“Proposal.log”}, \\
\quad \mathit{CurrState} \leftarrow [\\
\qquad \mathit{proposals} \mapsto \mathit{proposal}, \\
\qquad \mathit{configuration} \mapsto \mathit{configuration}, \\
\qquad \mathit{mastership} \mapsto \mathit{mastership}, \\
\qquad \mathit{target} \mapsto \mathit{target}], \\
\quad \mathit{SuccState} \leftarrow [\\
\qquad \mathit{proposals} \mapsto \mathit{proposal}', \\
\qquad \mathit{configuration} \mapsto \mathit{configuration}', \\
\qquad \mathit{mastership} \mapsto \mathit{mastership}', \\
\qquad \mathit{target} \mapsto \mathit{target}']
\end{array}
$$

$$
\text{LOCAL } \mathit{Max}(s) \triangleq \text{CHOOSE } i \in s : \forall\, j \in s : i > j
$$

$$
\begin{array}{l}
\mathit{CommitChange}(n,\ i) \triangleq \\
\quad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{phase} = \mathit{Commit} \\
\quad \land\ \mathit{proposal}[i].\mathit{change}.\mathit{state}\ = \mathit{InProgress} \\
\qquad \text{If the committed index does not match the proposal index, commit the change.} \\
\quad \land\ \lor\ \land\ \mathit{configuration}.\mathit{committed}.\mathit{index} = i - 1 \\
\qquad\qquad \text{If the change is valid, update the committed index, revision, and values.} \\
\qquad\qquad \land\ \lor\ \mathit{configuration}' = [\mathit{configuration} \text{ EXCEPT } !.\mathit{committed}.\mathit{index}\ = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !.\mathit{committed}.\mathit{revision}\ = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !.\mathit{committed}.\mathit{values}\ = \mathit{proposal}[i].\mathit{change}.\mathit{values}\ @@ \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{configuration}.\mathit{committed}.\mathit{valu}
\end{array}
$$

```
                If the change is invalid, update only the committed index.
              ∨ configuration' = [configuration EXCEPT !.committed.index = i]
          ∧ UNCHANGED ⟨proposal⟩
        If both the committed index and committed revision were updated, the proposal was successful.
      ∨ ∧ configuration.committed.index = i
        ∧ configuration.committed.revision = i
        ∧ proposal' = [proposal EXCEPT ![i].change.state = Complete]
        ∧ UNCHANGED ⟨configuration⟩
        If the committed index was updated but the revision was not, the proposal failed validation.
      ∨ ∧ configuration.committed.index = i
        ∧ configuration.committed.revision ≠ i
        ∧ proposal' = [proposal EXCEPT ![i].change.state = Failed]
        ∧ UNCHANGED ⟨configuration⟩
    ∧ UNCHANGED ⟨target⟩

ApplyChange(n, i) ≜
    ∧ proposal[i].change.phase = Apply
    ∧ proposal[i].change.state  = InProgress
        If the applied index does not match the proposal index, apply the change.
    ∧ ∨ ∧ configuration.applied.index = i − 1
        ∧ configuration.state = Complete
        ∧ configuration.term = mastership.term
            If the change can be applied, update the index, revision, and values.
        ∧ ∨ ∧ target' = [target EXCEPT !.values = proposal[i].change.values @@ target.values]
            ∧ configuration' = [configuration EXCEPT !.applied.index    = i,
                                                     !.applied.revision = i,
                                                     !.applied.values   = proposal[i].change.values @@
                                                                          configuration.applied.values]
                If the change is invalid, update only the applied index.
              ∨ configuration' = [configuration EXCEPT !.applied.index = i]
        ∧ UNCHANGED ⟨proposal⟩
        If the applied index and revision both match the proposal index, the change was successful.
      ∨ ∧ configuration.applied.index = i
        ∧ configuration.applied.revision = i
        ∧ proposal' = [proposal EXCEPT ![i].change.state = Complete]
        ∧ UNCHANGED ⟨configuration⟩
        If the applied index matches the proposal index but the revision does not, the proposal failed.
      ∨ ∧ configuration.applied.index = i
        ∧ configuration.applied.revision ≠ i
        ∧ proposal' = [proposal EXCEPT ![i].change.state = Failed]
        ∧ UNCHANGED ⟨configuration⟩
    ∧ UNCHANGED ⟨target⟩

CommitRollback(n, i) ≜
    ∧ proposal[i].rollback.phase = Commit
```

```
    ∧ proposal[i].rollback.state  = InProgress
        If the committed revision matches the proposal revision, roll back to the previous revision.
    ∧ ∨ ∧ configuration.committed.revision = i
        ∧ configuration' = [configuration EXCEPT !.committed.revision = proposal[i].rollback.revision,
                                                 !.committed.values   = proposal[i].rollback.values @@
                                                                        configuration.committed.values]
        ∧ UNCHANGED ⟨proposal⟩
        If the committed index matches the rollback index, complete the rollback.
      ∨ ∧ configuration.committed.revision = proposal[i].rollback.revision
        ∧ proposal' = [proposal EXCEPT ![i].rollback.state = Complete]
        ∧ UNCHANGED ⟨configuration⟩
    ∧ UNCHANGED ⟨target⟩

ApplyRollback(n, i) ≜
    ∧ proposal[i].rollback.phase = Apply
    ∧ proposal[i].rollback.phase = InProgress
        If the applied revision matches the proposal revision, roll back to the previous revision.
    ∧ ∨ ∧ configuration.applied.revision = i
        ∧ target' = [target EXCEPT !.values = proposal[i].rollback.values @@ target.values]
        ∧ configuration' = [configuration EXCEPT !.applied.revision = proposal[i].rollback.revision,
                                                 !.applied.values   = proposal[i].rollback.values @@
                                                                      configuration.applied.values]
        ∧ UNCHANGED ⟨proposal⟩
        If the committed index matches the rollback index, complete the rollback.
      ∨ ∧ configuration.committed.revision = proposal[i].rollback.revision
        ∧ proposal' = [proposal EXCEPT ![i].rollback.state = Complete]
        ∧ UNCHANGED ⟨configuration, target⟩

  Reconcile a proposal
ReconcileProposal(n, i) ≜
    ∧ i ∈ DOMAIN proposal
    ∧ mastership.master = n
    ∧ ∨ CommitChange(n, i)
      ∨ ApplyChange(n, i)
      ∨ CommitRollback(n, i)
      ∨ ApplyRollback(n, i)
    ∧ UNCHANGED ⟨mastership⟩
```