MODULE *Proposal*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant
CONSTANT *Nil*

Event constants
CONSTANTS
Change,
Rollback

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,
conn,
target

A record of per-target proposals

VARIABLE *proposal*

A sequence of configuration changes used for model checking.

VARIABLE *history*

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

$$
\begin{array}{l}
\text{LOCAL } \mathit{State} \triangleq [\\
\quad \mathit{proposals} \mapsto [i \in \text{DOMAIN } \mathit{proposal} \mapsto \mathit{proposal}[i] \mathbin{@@} [\mathit{index} \mapsto i]], \\
\quad \mathit{configuration} \mapsto \mathit{configuration}, \\
\quad \mathit{mastership} \mapsto \mathit{mastership}, \\
\quad \mathit{conns} \mapsto \mathit{conn}, \\
\quad \mathit{target} \mapsto \mathit{target}]
\end{array}
$$

$$
\begin{array}{l}
\text{LOCAL } \mathit{Transitions} \triangleq \\
\quad \text{LET} \\
\qquad \mathit{proposals} \triangleq \{i \in \text{DOMAIN } \mathit{proposal}' : \\
\qquad\qquad\qquad i \in \text{DOMAIN } \mathit{proposal} \Rightarrow \mathit{proposal}'[i] \neq \mathit{proposal}[i]\} \\
\quad \text{IN} \\
\qquad [\mathit{proposals} \mapsto [i \in \mathit{proposals} \mapsto \mathit{proposal}'[i] \mathbin{@@} [\mathit{index} \mapsto i]]] \mathbin{@@} \\
\qquad (\text{IF } \mathit{configuration}' \neq \mathit{configuration} \text{ THEN } [\mathit{configuration} \mapsto \mathit{configuration}'] \text{ ELSE } \langle\rangle) \mathbin{@@} \\
\qquad (\text{IF } \mathit{target}' \neq \mathit{target} \text{ THEN } [\mathit{target} \mapsto \mathit{target}'] \text{ ELSE } \langle\rangle)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Test} \triangleq \text{INSTANCE } \mathit{Test} \text{ WITH} \\
\quad \mathit{File} \leftarrow \text{“Proposal.log”}
\end{array}
$$

$$
\begin{array}{l}
\textit{CommitChange}(n,\ i) \triangleq \\
\quad \land\ \textit{proposal}[i].\textit{change.phase} = \textit{Commit} \\
\quad \land\ \textit{proposal}[i].\textit{change.state} = \textit{InProgress} \\
\qquad \text{If the committed index does not match the proposal index, commit the change.} \\
\quad \land\ \lor\ \land\ \textit{configuration.committed.index} = i - 1 \\
\qquad\qquad \text{If the change is valid, update the committed index, revision, and values.} \\
\qquad\qquad \land\ \lor\ \land\ \textit{configuration}' = [\textit{configuration}\ \text{EXCEPT}\ !.\textit{committed.index} = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad !.\textit{committed.revision} = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad !.\textit{committed.values} = \textit{proposal}[i].\textit{change.values} \mathbin{@@} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{configuration.committed.va} \\
\qquad\qquad\qquad\ \land\ \textit{history}' = \textit{Append}(\textit{history}, [\textit{type} \mapsto \textit{Change}, \textit{phase} \mapsto \textit{Commit}, \textit{index} \mapsto i]) \\
\qquad\qquad\qquad \text{If the change is invalid, update only the committed index.} \\
\qquad\qquad\qquad\lor\ \land\ \textit{configuration}' = [\textit{configuration}\ \text{EXCEPT}\ !.\textit{committed.index} = i] \\
\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle \textit{history} \rangle \\
\qquad\qquad \land\ \text{UNCHANGED}\ \langle \textit{proposal} \rangle \\
\qquad\qquad \text{If both the committed index and committed revision were updated, the proposal was successful.} \\
\qquad\qquad \lor\ \land\ \textit{configuration.committed.index} = i \\
\qquad\qquad\qquad\ \land\ \textit{configuration.committed.revision} = i \\
\qquad\qquad\qquad\ \land\ \textit{proposal}' = [\textit{proposal}\ \text{EXCEPT}\ ![i].\textit{change.state} = \textit{Complete}] \\
\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle \textit{configuration}, \textit{history} \rangle \\
\qquad\qquad \text{If the committed index was updated but the revision was not, the proposal failed validation.} \\
\qquad\qquad \lor\ \land\ \textit{configuration.committed.index} = i \\
\qquad\qquad\qquad\ \land\ \textit{configuration.committed.revision} \neq i \\
\qquad\qquad\qquad\ \land\ \textit{proposal}' = [\textit{proposal}\ \text{EXCEPT}\ ![i].\textit{change.state} = \textit{Failed}] \\
\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle \textit{configuration}, \textit{history} \rangle \\
\quad \land\ \text{UNCHANGED}\ \langle \textit{target} \rangle
\end{array}
$$

$$
\begin{array}{l}
\textit{ApplyChange}(n,\ i) \triangleq \\
\quad \land\ \textit{proposal}[i].\textit{change.phase} = \textit{Apply} \\
\quad \land\ \textit{proposal}[i].\textit{change.state} = \textit{InProgress} \\
\qquad \text{If the applied index does not match the proposal index, apply the change.} \\
\quad \land\ \lor\ \land\ \textit{configuration.applied.index} = i - 1 \\
\qquad\qquad \land\ \textit{configuration.state} = \textit{Complete} \\
\qquad\qquad \land\ \textit{configuration.term} = \textit{mastership.term} \\
\qquad\qquad \land\ \textit{conn}[n].\textit{id} = \textit{mastership.conn} \\
\qquad\qquad \land\ \textit{conn}[n].\textit{connected} \\
\qquad\qquad \land\ \textit{target.running} \\
\qquad\qquad\qquad \text{If the change can be applied, update the index, revision, and values.} \\
\qquad\qquad \land\ \lor\ \land\ \textit{target}' = [\textit{target}\ \text{EXCEPT}\ !.\textit{values} = \textit{proposal}[i].\textit{change.values} \mathbin{@@} \textit{target.values}] \\
\qquad\qquad\qquad\ \land\ \textit{configuration}' = [\textit{configuration}\ \text{EXCEPT}\ !.\textit{applied.index} = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad !.\textit{applied.revision} = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad !.\textit{applied.values} = \textit{proposal}[i].\textit{change.values} \mathbin{@@} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{configuration.applied.values}]
\end{array}
$$

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

If the change is invalid, update only the applied index.

$\lor\ \land\ configuration' = [configuration\ \text{EXCEPT}\ !.applied.index = i]$
$\land\ \text{UNCHANGED}\ \langle target,\ history\rangle$

$\land\ \text{UNCHANGED}\ \langle proposal\rangle$

If the applied index and revision both match the proposal index, the change was successful.

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

If the applied index matches the proposal index but the revision does not, the proposal failed.

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

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

$\land\ proposal[i].rollback.phase = Commit$
$\land\ proposal[i].rollback.state\ = InProgress$

If the committed revision matches the proposal revision, roll back to the previous revision.

$\land\ \lor\ \land\ configuration.committed.revision = i$
$\land\ configuration' = [configuration\ \text{EXCEPT}\ !.committed.revision = proposal[i].rollback.revision,$
$!.committed.values\ = proposal[i].rollback.values\ @@$
$configuration.committed.values]$
$\land\ history' = Append(history,\ [type \mapsto Rollback,\ phase \mapsto Commit,\ index \mapsto i])$
$\land\ \text{UNCHANGED}\ \langle proposal\rangle$

If the committed index matches the rollback index, complete the rollback.

$\lor\ \land\ configuration.committed.revision = proposal[i].rollback.revision$
$\land\ proposal' = [proposal\ \text{EXCEPT}\ ![i].rollback.state = Complete]$
$\land\ \text{UNCHANGED}\ \langle configuration,\ history\rangle$

$\land\ \text{UNCHANGED}\ \langle target\rangle$

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

$\land\ proposal[i].rollback.phase = Apply$
$\land\ proposal[i].rollback.state\ = InProgress$

If the applied revision matches the proposal revision, roll back to the previous revision.

$\land\ \lor\ \land\ configuration.applied.revision = i$
$\land\ configuration.state = Complete$
$\land\ configuration.term = mastership.term$
$\land\ conn[n].id = mastership.conn$
$\land\ conn[n].connected$
$\land\ target.running$
$\land\ target' = [target\ \text{EXCEPT}\ !.values = proposal[i].rollback.values\ @@\ target.values]$
$\land\ configuration' = [configuration\ \text{EXCEPT}\ !.applied.revision = proposal[i].rollback.revision,$
$!.applied.values\ = proposal[i].rollback.values\ @@$

```
                                                    configuration.applied.values]
        ∧ history' = Append(history, [type ↦ Rollback, phase ↦ Apply, index ↦ i])
        ∧ 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, history⟩

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, conn⟩
```