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*

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

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

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

$$\mathit{CommitChange}(n,\ i) \triangleq$$

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

If the committed index does not match the proposal index, commit the change.

$\land\ \lor\ \land\ configuration.committed.index = i - 1$

If the change is valid, update the committed index, revision, and values.

$\land\ \lor\ \land\ configuration' = [configuration\ \text{EXCEPT}\ !.committed.index\ = i,$
$!.committed.revision\ = i,$
$!.committed.values\ = proposal[i].change.values\ @@$
$configuration.committed.va$
$\land\ history' = Append(history, [type \mapsto Change, phase \mapsto Commit, index \mapsto i])$

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

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

If both the committed index and committed revision were updated, the proposal was successful.

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

If the committed index was updated but the revision was not, the proposal failed validation.

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

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

$ApplyChange(n, i)\ \triangleq$
$\land\ proposal[i].change.phase = Apply$
$\land\ proposal[i].change.state\ = InProgress$

If the applied index does not match the proposal index, apply the change.

$\land\ \lor\ \land\ configuration.applied.index = i - 1$
$\land\ configuration.state = Complete$
$\land\ configuration.term = mastership.term$
$\land\ conn[n].id = mastership.conn$
$\land\ conn[n].connected$
$\land\ target.running$

If the change can be applied, update the index, revision, and values.

$\land\ \lor\ \land\ target' = [target\ \text{EXCEPT}\ !.values = proposal[i].change.values\ @@\ target.values]$
$\land\ configuration' = [configuration\ \text{EXCEPT}\ !.applied.index\ = i,$
$!.applied.revision\ = i,$
$!.applied.values\ = proposal[i].change.values\ @@$
$configuration.applied.values]$
$\land\ history' = Append(history, [type \mapsto Change, phase \mapsto Apply, index \mapsto i])$

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

$\lor\ \land\ configuration' = [configuration\ \text{EXCEPT}\ !.applied.index = i]$

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

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

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

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

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

$\lor \land configuration.applied.index = i$
$\land configuration.applied.revision \neq i$
$\land proposal' = [proposal$ EXCEPT $![i].change.state = Failed]$
$\land$ 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.

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

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

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

$\land$ 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.

$\land \lor \land configuration.applied.revision = i$
$\land configuration.state = Complete$
$\land configuration.term = mastership.term$
$\land conn[n].id = mastership.conn$
$\land conn[n].connected$
$\land target.running$
$\land target' = [target$ EXCEPT $!.values = proposal[i].rollback.values \mathbin{@@} target.values]$
$\land configuration' = [configuration$ EXCEPT $!.applied.revision = proposal[i].rollback.revision,$
$!.applied.values = proposal[i].rollback.values \mathbin{@@} configuration.applied.values]$
$\land history' = Append(history, [type \mapsto Rollback, phase \mapsto Apply, index \mapsto i])$
$\land$ UNCHANGED $\langle proposal \rangle$

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

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

Reconcile a proposal

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