MODULE *Proposal*

EXTENDS *Configuration*, *Mastership*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

CONSTANT *NumProposals*

ASSUME $NumProposals \in Nat$

Transaction type constants
CONSTANTS
ProposalChange,
ProposalRollback

Phase constants
CONSTANTS
ProposalCommit,
ProposalApply

Status constants
CONSTANTS
ProposalPending,
ProposalInProgress,
ProposalComplete,
ProposalFailed

CONSTANT *LogProposal*

ASSUME $LogProposal \in$ BOOLEAN

A record of per-target proposals
VARIABLE *proposal*

LOCAL $CurrentState \triangleq [$
$proposals \mapsto [i \in \{i \in \text{DOMAIN } proposal : proposal[i].state \neq Nil\} \mapsto proposal[i]],$
$configuration \mapsto configuration,$
$target \mapsto target,$
$mastership \mapsto mastership,$
$nodes \mapsto node]$

LOCAL $SuccessorState \triangleq [$

$proposals \mapsto [i \in \{i \in \text{DOMAIN } proposal' : proposal'[i].state \neq Nil\} \mapsto proposal'[i]],$
$configuration \mapsto configuration',$
$target \mapsto target',$
$mastership \mapsto mastership',$
$nodes \mapsto node']$

LOCAL $Log \triangleq$ INSTANCE Log WITH
$File \leftarrow$ "Proposal.log",
$CurrentState \leftarrow CurrentState,$
$SuccessorState \leftarrow SuccessorState,$
$Enabled \leftarrow LogProposal$

Commit a change to the configuration.
A change can be committed once all prior changes have been committed.
If a prior change is being rolled back, the rollback must complete
before the change can be committed. Changes must be committed in
sequential order.
Once a change commit is in progress, the change must be committed or
failed before it can be applied or rolled back.

$CommitChange(n,\ i) \triangleq$
$\land \lor \land proposal[i].change.status = ProposalPending$
To commit a change, the commit index must be the prior index,
and the target commit index must match the commit revision.
$\land configuration.commit.index = i - 1$
$\land configuration.commit.revision = configuration.commit.target$
$\land configuration' = [configuration \text{ EXCEPT } !.commit.index = i,$
$!.commit.target = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalInProgress]$
$\lor \land proposal[i].change.status = ProposalInProgress$
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.
$\land \lor \text{LET } rollbackRevision \triangleq configuration.commit.revision$
$rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
IF $p \in \text{DOMAIN } configuration.commit.values$ THEN
$configuration.commit.values[p]$
ELSE
$[index \mapsto 0,\ value \mapsto Nil]]$
$changeValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
$proposal[i].change.values[p] \mathbin{@@} [index \mapsto i]]$
IN $\land configuration' = [configuration \text{ EXCEPT } !.commit.revision = i,$
$!.commit.values = changeValues]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.values = changeValues,$
$![i].change.phase = ProposalApply,$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !\,[i].change.status \quad\;\; = ProposalPending, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !\,[i].rollback.revision \;= rollbackRevision, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !\,[i].rollback.values \quad\;\; = rollbackValues] \\
\qquad\qquad\quad \lor\ \land\ configuration' = [configuration \text{ EXCEPT } !.commit.revision = i] \\
\qquad\qquad\qquad\ \land\ proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalFailed] \\
\qquad\ \land \text{ UNCHANGED } \langle target \rangle
\end{array}
$$

Apply a change to the target.
A change can be applied once all prior changes have been applied.
If a prior change failed being applied, it must be rolled back before
any subsequent change can be applied.

$$
\begin{array}{l}
ApplyChange(n,\ i) \triangleq \\
\qquad \land\ \lor\ \land\ proposal[i].change.status = ProposalPending \\
\qquad\qquad \text{To apply a change, the apply index must be the prior index.} \\
\qquad\qquad\ \land\ configuration.apply.index = i - 1 \\
\qquad\qquad\ \land\ configuration.apply.revision = configuration.apply.target \\
\qquad\qquad\ \land\ configuration' = [configuration \text{ EXCEPT } !.apply.index \;= i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !.apply.target \;= i] \\
\qquad\qquad\ \land\ proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalInProgress] \\
\qquad\qquad\ \land \text{ UNCHANGED } \langle target \rangle \\
\qquad\quad \lor\ \land\ proposal[i].change.status = ProposalInProgress \\
\qquad\qquad \text{Verify the applied term is the current } \textit{mastership} \text{ term to ensure the} \\
\qquad\qquad \text{configuration has been synchronized following restarts.} \\
\qquad\qquad\ \land\ configuration.apply.term = mastership.term \\
\qquad\qquad \text{Verify the node's connection to the target.} \\
\qquad\qquad\ \land\ node[n].connected \\
\qquad\qquad\ \land\ mastership.conn = node[n].incarnation \\
\qquad\qquad\ \land\ target.running \\
\qquad\qquad\ \land\ node[n].target = target.incarnation \\
\qquad\qquad \text{Model successful and failed target update requests.} \\
\qquad\qquad\ \land\ \lor\ \land\ target' = [target \text{ EXCEPT } !.values = proposal[i].change.values \mathbin{@@} target.values] \\
\qquad\qquad\qquad\qquad\ \land \text{ LET } values \triangleq proposal[i].change.values \mathbin{@@} configuration.apply.values \\
\qquad\qquad\qquad\qquad\qquad \text{IN } \quad configuration' = [configuration \text{ EXCEPT } !.apply.revision \;\;\;\;\;\; = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !.apply.incarnation = target.incarnation, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad !.apply.values \;\;\;\;\;\; = values] \\
\qquad\qquad\qquad\qquad\ \land\ proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalComplete] \\
\qquad\qquad\qquad\quad \text{If the proposal could not be applied, mark it failed but do not update the} \\
\qquad\qquad\qquad\quad \text{last applied index. The proposal must be rolled back before new proposals} \\
\qquad\qquad\qquad\quad \text{can be applied to the configuration/target.} \\
\qquad\qquad\qquad\ \lor\ \land\ proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalFailed] \\
\qquad\qquad\qquad\qquad\ \land \text{ UNCHANGED } \langle configuration,\ target \rangle
\end{array}
$$

Commit a rollback to the configuration.
A change can be rolled back once all subsequent, non-pending changes have been
rolled back.

$CommitRollback(n,\ i) \triangleq$

$\land \lor \land proposal[i].rollback.status = ProposalPending$
$\land configuration.commit.target = i$
$\land configuration.commit.revision = i$
$\land configuration' = [configuration \text{ EXCEPT } !.commit.target = proposal[i].rollback.revision]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].rollback.status = ProposalInProgress]$
$\lor \land proposal[i].rollback.status = ProposalInProgress$
$\land \text{LET } revision \triangleq proposal[i].rollback.revision$
$values \triangleq proposal[i].rollback.values$
IN $\land configuration' = [configuration \text{ EXCEPT } !.commit.revision = revision,$
$!.commit.values = values]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].rollback.phase = ProposalApply,$
$![i].rollback.status = ProposalPending]$
$\land \text{UNCHANGED } \langle target \rangle$

Commit a rollback to the target.

A change can be rolled back once all subsequent, non-pending changes have been rolled back.

$ApplyRollback(n,\ i) \triangleq$

$\land \lor \land proposal[i].rollback.status = ProposalPending$
$\land configuration.apply.target = i$
$\land configuration.apply.revision = i$
$\land configuration' = [configuration \text{ EXCEPT } !.apply.target = proposal[i].rollback.revision]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].rollback.status = ProposalInProgress]$
$\land \text{UNCHANGED } \langle target \rangle$
$\lor \land proposal[i].rollback.status = ProposalInProgress$

Verify the applied term is the current *mastership* term to ensure the configuration has been synchronized following restarts.

$\land configuration.apply.term = mastership.term$

Verify the node's connection to the target.

$\land node[n].connected$
$\land target.running$
$\land target' = [target \text{ EXCEPT } !.values = proposal[i].rollback.values \mathbin{@@} target.values]$
$\land \text{LET } revision \triangleq proposal[i].rollback.revision$
$values \triangleq proposal[i].rollback.values \mathbin{@@} configuration.apply.values$
IN $configuration' = [configuration \text{ EXCEPT } !.apply.revision = revision,$
$!.apply.values = values]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].rollback.status = ProposalComplete]$

$ReconcileProposal(n,\ i) \triangleq$

$\land mastership.master = n$
$\land \lor \land proposal[i].state = ProposalChange$
$\land \lor \land proposal[i].change.phase = ProposalCommit$
$\land CommitChange(n,\ i)$
$\lor \land proposal[i].change.phase = ProposalApply$

$$
\begin{array}{l}
\quad\quad\quad \land\ ApplyChange(n,\ i) \\
\quad\quad \lor\ \land\ proposal[i].state = ProposalRollback \\
\quad\quad\quad \land\ \lor\ \land\ proposal[i].rollback.phase = ProposalCommit \\
\quad\quad\quad\quad\quad \land\ CommitRollback(n,\ i) \\
\quad\quad\quad\quad \lor\ \land\ proposal[i].rollback.phase = ProposalApply \\
\quad\quad\quad\quad\quad \land\ ApplyRollback(n,\ i) \\
\quad \land \text{ UNCHANGED } \langle mastership,\ node \rangle
\end{array}
$$

Formal specification, constraints, and theorems.

$$
\begin{array}{l}
InitProposal\ \triangleq \\
\quad \land\ Log\,!\,Init \\
\quad \land\ proposal = [\\
\quad\quad\quad i \in 1\ ..\ NumProposals \mapsto [\\
\quad\quad\quad\quad state \quad\quad \mapsto Nil, \\
\quad\quad\quad\quad change \quad \mapsto [\\
\quad\quad\quad\quad\quad values \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto Nil]], \\
\quad\quad\quad\quad\quad phase \ \mapsto Nil, \\
\quad\quad\quad\quad\quad status \mapsto Nil], \\
\quad\quad\quad\quad rollback \mapsto [\\
\quad\quad\quad\quad\quad revision \mapsto 0, \\
\quad\quad\quad\quad\quad values \quad \mapsto [p \in \{\} \mapsto [index \mapsto 0,\ value \mapsto Nil]], \\
\quad\quad\quad\quad\quad phase \quad \mapsto Nil, \\
\quad\quad\quad\quad\quad status \quad \mapsto Nil]]]
\end{array}
$$

$$
\begin{array}{l}
NextProposal\ \triangleq \\
\quad \lor\ \exists\, n \in Nodes : \\
\quad\quad\ \exists\, i \in \text{DOMAIN } proposal : \\
\quad\quad\quad Log\,!\,Action(ReconcileProposal(n,\ i),\ [node \mapsto n,\ index \mapsto i])
\end{array}
$$

---

* Modification History
* Last modified *Fri Apr* 21 19:15:11 *PDT* 2023 by *jhalterm*
* Last modified *Mon Feb* 21 01:24:12 *PST* 2022 by *jordanhalterman*
* Created Sun *Feb* 20 10:07:16 *PST* 2022 by *jordanhalterman*