MODULE *Proposal*

EXTENDS *Configuration*, *Mastership*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

LOCAL INSTANCE *TLCExt*

CONSTANT *NumProposals*

ASSUME $NumProposals \in Nat$

Transaction type constants
CONSTANTS
ProposalChange,
ProposalRollback

Phase constants
CONSTANTS
ProposalCommit,
ProposalApply

Status constants
CONSTANTS
ProposalPending,
ProposalInProgress,
ProposalComplete,
ProposalAborted,
ProposalFailed

CONSTANT *LogProposal*

ASSUME $LogProposal \in$ BOOLEAN

A record of per-target proposals
VARIABLE *proposal*

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

$nodes \mapsto node]$

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

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

$ProposalDone \triangleq \{ProposalComplete, ProposalAborted, ProposalFailed\}$

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.

$CommitChange(n, i) \triangleq$
$\land \lor \land proposal[i].change.commit = ProposalPending$
To apply a change, the prior change must have been committed. Additionally,
the configuration's applied index must match the proposed index to prevent
commits while a prior change is still being rolled back.
$\land i - 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].change.commit \in ProposalDone$
$\land configuration.commit.index = configuration.commit.proposal$
$\land \lor \land proposal[i].rollback.commit \neq ProposalPending$
$\land configuration' = [configuration \text{ EXCEPT } !.commit.proposal = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = ProposalInProgress,$
$![i].rollback.index = configuration.commit.index]$
$\lor \land proposal[i].rollback.commit = ProposalPending$
$\land configuration' = [configuration \text{ EXCEPT } !.commit.proposal = i,$
$!.commit.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = ProposalAborted,$
$![i].rollback.index = configuration.commit.index]$
$\lor \land proposal[i].change.commit = ProposalInProgress$
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.

$\land \lor$ LET $rollbackIndex \triangleq configuration.commit.index$
$rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
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 $\land configuration' = [configuration \text{ EXCEPT } !.commit.index = i,$
$!.commit.values = changeValues]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.values = changeValues,$
$![i].rollback.values = rollbackValues,$
$![i].change.commit = ProposalComplete]$
$\lor \land configuration' = [configuration \text{ EXCEPT } !.commit.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.commit = ProposalFailed]$
$\land$ UNCHANGED $\langle target \rangle$

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.

$ApplyChange(n,\ i) \triangleq$
$\land \lor \land proposal[i].change.apply = ProposalPending$
To apply a change, the change must have been committed and the prior
change applied. Additionally, the configuration's applied index must
match the proposed index to prevent applies while a prior change is
still being rolled back.
$\land i - 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i-1].change.apply \in ProposalDone$
$\land configuration.apply.index = configuration.apply.proposal$
The change cannot be applied until the commit is complete.
$\land \lor \land proposal[i].change.commit = ProposalComplete$
$\land \lor \land proposal[i].rollback.apply \neq ProposalPending$
$\land configuration' = [configuration \text{ EXCEPT } !.apply.proposal = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = ProposalInProgress]$
$\lor \land proposal[i].rollback.apply = ProposalPending$
$\land configuration' = [configuration \text{ EXCEPT } !.apply.proposal = i,$
$!.apply.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = ProposalAborted]$
$\lor \land proposal[i].change.commit \in \{ProposalAborted,\ ProposalFailed\}$
$\land configuration' = [configuration \text{ EXCEPT } !.apply.proposal = i,$
$!.apply.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.apply = ProposalAborted]$
$\land$ UNCHANGED $\langle target \rangle$
$\lor \land proposal[i].change.apply = ProposalInProgress$
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.

$\land\ node[n].connected$

$\land\ mastership.conn = node[n].incarnation$

$\land\ target.running$

$\land\ node[n].target = target.incarnation$

Model successful and failed target update requests.

$\land\ \lor\ \land\ target' = [target \text{ EXCEPT } !.values = proposal[i].change.values \mathbin{@@} target.values]$

$\land\ \text{LET } values \triangleq proposal[i].change.values \mathbin{@@} configuration.apply.values$

$\text{IN } configuration' = [configuration \text{ EXCEPT } !.apply.index = i,$

$!.apply.target = target.incarnation,$

$!.apply.values = values]$

$\land\ proposal' = [proposal \text{ EXCEPT } ![i].change.apply = ProposalComplete]$

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.

$\lor\ \land\ proposal' = [proposal \text{ EXCEPT } ![i].change.apply = ProposalFailed]$

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

Commit a rollback to the configuration.

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

$CommitRollback(n, i) \triangleq$

$\land\ \lor\ \land\ proposal[i].rollback.commit = ProposalPending$

$\land\ configuration.commit.proposal = i$

$\land\ configuration.commit.index = i$

$\land\ i + 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i + 1].rollback.commit = ProposalComplete$

If the change is committed, it cannot be rolled back until both the commit proposal and the commit index match the proposal's index. This ensures this is the proposal is the last committed to the configuration.

$\land\ \lor\ \land\ proposal[i].change.commit = ProposalComplete$

$\land\ configuration' = [configuration \text{ EXCEPT } !.commit.proposal = proposal[i].rollback.index]$

$\land\ proposal' = [proposal \text{ EXCEPT } ![i].rollback.commit = ProposalInProgress]$

If the change commit failed, we still have to wait until the commit proposal and index match this proposal index, but we can complete the rollback directly from there since it failed validation and therefore was never applied to the configuration.

$\lor\ \land\ proposal[i].change.commit = ProposalFailed$

$\land\ configuration' = [configuration \text{ EXCEPT } !.commit.proposal = proposal[i].rollback.index,$

$!.commit.index = proposal[i].rollback.index]$

$\land\ proposal' = [proposal \text{ EXCEPT } ![i].rollback.commit = ProposalComplete]$

If the change commit was aborted, the rollback can be completed once the configuration's commit proposal and index match this proposal's rollback

index, indicating all subsequent changes have been rolled back.

$\lor \land proposal[i].change.commit = ProposalAborted$
$\quad \land configuration' = [configuration \text{ EXCEPT } !.commit.proposal = proposal[i].rollback.index,$
$\qquad\qquad !.commit.index = proposal[i].rollback.index]$
$\quad \land proposal' = [proposal \text{ EXCEPT } ![i].rollback.commit = ProposalComplete]$
$\land \text{UNCHANGED } \langle target \rangle$
$\lor \land proposal[i].rollback.commit = ProposalInProgress$
$\quad \land \text{LET } index \triangleq proposal[i].rollback.index$
$\qquad values \triangleq proposal[i].rollback.values \mathbin{@@} configuration.commit.values$
$\qquad \text{IN} \quad \land configuration' = [configuration \text{ EXCEPT } !.commit.index = index,$
$\qquad\qquad\qquad\qquad !.commit.values = values]$
$\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![i].rollback.commit = ProposalComplete]$
$\land \text{UNCHANGED } \langle target \rangle$

Commit a rollback to the target.

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

$ApplyRollback(n,\ i) \triangleq$
$\land \lor \land proposal[i].rollback.apply = ProposalPending$
$\quad \land i + 1 \in \text{DOMAIN } proposal \Rightarrow proposal[i + 1].rollback.apply = ProposalComplete$

The rollback cannot be applied until the commit is complete.

$\quad \land proposal[i].rollback.commit = ProposalComplete$

If the change is applied, it cannot be rolled back until both the apply proposal and the apply index match the proposal's index. This ensures this is the proposal is the last applied to the configuration.

$\land \lor \land proposal[i].change.apply = ProposalComplete$
$\quad \land configuration.apply.proposal = i$
$\quad \land configuration.apply.index = i$
$\quad \land configuration' = [configuration \text{ EXCEPT } !.apply.proposal = proposal[i].rollback.index]$
$\quad \land proposal' = [proposal \text{ EXCEPT } ![i].rollback.apply = ProposalInProgress]$

Rollbacks must be applied for failures to account for races that may or may not have resulted in changes applying to a target.

If the change commit failed, we have to wait until the applied proposal patches this proposal index and the applied index matches this proposal's rollback index before transitioning to the in-progress state. Update the configuration's applied proposal to the rollback index and the applied index to this proposal's index to block other proposals until this rollback is complete.

$\lor \land proposal[i].change.apply = ProposalFailed$
$\quad \land configuration.apply.proposal = i$
$\quad \land configuration.apply.index = proposal[i].rollback.index$
$\quad \land configuration' = [configuration \text{ EXCEPT } !.apply.proposal = proposal[i].rollback.index,$
$\qquad\qquad !.apply.index = i]$
$\quad \land proposal' = [proposal \text{ EXCEPT } ![i].rollback.apply = ProposalInProgress]$

```
                If the change apply was aborted, the rollback can be completed once the
                configuration's applied proposal and index match this proposal's rollback
                index, indicating all subsequent changes have been rolled back.
              ∨ ∧ proposal[i].change.apply = ProposalAborted
                ∧ configuration.apply.proposal = i
                ∧ configuration.apply.index = i
                ∧ configuration' = [configuration EXCEPT !.apply.proposal = proposal[i].rollback.index,
                                                         !.apply.index    = proposal[i].rollback.index]
                ∧ proposal' = [proposal EXCEPT ![i].rollback.apply = ProposalComplete]
            ∧ UNCHANGED ⟨target⟩
        ∨ ∧ proposal[i].rollback.apply = ProposalInProgress
            Verify the applied term is the current mastership term to ensure the
            configuration has been synchronized following restarts.
          ∧ configuration.apply.term = mastership.term
            Verify the node's connection to the target.
          ∧ node[n].connected
          ∧ target.running
          ∧ target' = [target EXCEPT !.values = proposal[i].rollback.values @@ target.values]
          ∧ LET index  ≜ proposal[i].rollback.index
                values ≜ proposal[i].rollback.values @@ configuration.apply.values
            IN  configuration' = [configuration EXCEPT !.apply.index  = index,
                                                       !.apply.values = values]
          ∧ proposal' = [proposal EXCEPT ![i].rollback.apply = ProposalComplete]

ReconcileProposal(n, i) ≜
  ∧ mastership.master = n
  ∧ ∨ ∧ proposal[i].change.commit ∉ ProposalDone
      ∧ CommitChange(n, i)
    ∨ ∧ proposal[i].change.apply ∉ ProposalDone
      ∧ ApplyChange(n, i)
    ∨ ∧ proposal[i].rollback.commit ∉ ProposalDone
      ∧ CommitRollback(n, i)
    ∨ ∧ proposal[i].rollback.apply ∉ ProposalDone
      ∧ ApplyRollback(n, i)
  ∧ UNCHANGED ⟨mastership, node⟩

================================================================================

Formal specification, constraints, and theorems.
InitProposal ≜
  ∧ Log!Init
  ∧ proposal = [
      i ∈ 1 .. NumProposals ↦ [
        phase    ↦ Nil,
        change   ↦ [
          values  ↦ [p ∈ {} ↦ [index ↦ 0, value ↦ Nil]],
```

$$
\begin{array}{l}
\quad\quad\quad\quad \mathit{commit} \mapsto \mathit{Nil}, \\
\quad\quad\quad\quad \mathit{apply} \quad \mapsto \mathit{Nil}], \\
\quad\quad\quad \mathit{rollback} \mapsto [\\
\quad\quad\quad\quad \mathit{index} \quad \mapsto 0, \\
\quad\quad\quad\quad \mathit{values} \quad \mapsto [p \in \{\} \mapsto [\mathit{index} \mapsto 0, \; \mathit{value} \mapsto \mathit{Nil}]], \\
\quad\quad\quad\quad \mathit{commit} \mapsto \mathit{Nil}, \\
\quad\quad\quad\quad \mathit{apply} \quad \mapsto \mathit{Nil}]]]
\end{array}
$$

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

* 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*