MODULE *Proposals*

EXTENDS *Configurations*, *Mastership*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

Transaction type constants

CONSTANTS
ProposalChange,
ProposalRollback

Phase constants

CONSTANTS
ProposalInitialize,
ProposalValidate,
ProposalAbort,
ProposalCommit,
ProposalApply

Status constants

CONSTANTS
ProposalInProgress,
ProposalComplete,
ProposalFailed

A record of per-target proposals

VARIABLE *proposal*

LOCAL *InitState* $\triangleq$ [
proposals $\mapsto$ *proposal*,
configurations $\mapsto$ *configuration*,
targets $\mapsto$ *target*,
masterships $\mapsto$ *mastership*]

LOCAL *NextState* $\triangleq$ [
proposals $\mapsto$ *proposal*$'$,
configurations $\mapsto$ *configuration*$'$,
targets $\mapsto$ *target*$'$,
masterships $\mapsto$ *mastership*$'$]

LOCAL *Trace* $\triangleq$ INSTANCE *Trace* WITH

$Module \leftarrow$ "Proposals",
$InitState \leftarrow InitState$,
$NextState \leftarrow NextState$

Reconcile a proposal

$$
\begin{array}{l}
ReconcileProposal(n,\ i) \triangleq \\
\quad \land \lor \land proposal[i].phase = ProposalInitialize \\
\qquad \land \lor \land proposal[i].state = ProposalInProgress \\
\qquad\quad \land proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete] \\
\qquad\quad \land configuration' = [configuration \text{ EXCEPT } !.proposed.index = i] \\
\qquad\quad \land \text{UNCHANGED } \langle target \rangle \\
\qquad\ \lor \land proposal[i].state = ProposalComplete \\
\qquad\quad \land proposal' = [proposal \text{ EXCEPT } ![i].phase = ProposalValidate, \\
\qquad\qquad\qquad\qquad\qquad\qquad\ ![i].state\ = ProposalInProgress] \\
\qquad\quad \land \text{UNCHANGED } \langle configuration,\ target \rangle \\
\qquad \text{While in the Validate phase, validate the proposed changes.} \\
\qquad \text{If validation is successful, the proposal also records the changes} \\
\qquad \text{required to roll back the proposal and the index to which to roll back.} \\
\quad \lor \land proposal[i].phase = ProposalValidate \\
\qquad \land \lor \land proposal[i].state = ProposalInProgress \\
\qquad\quad \land configuration.index = i - 1 \\
\qquad\qquad \text{For Change proposals validate the set of requested changes.} \\
\qquad\quad \land \lor \land proposal[i].type = ProposalChange \\
\qquad\qquad \land \text{LET } rollbackIndex \triangleq configuration.committed.index \\
\qquad\qquad\qquad\quad rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto \\
\qquad\qquad\qquad\qquad\qquad \text{IF } p \in \text{DOMAIN } configuration.committed.values \text{ THEN} \\
\qquad\qquad\qquad\qquad\qquad\quad configuration.committed.values[p] \\
\qquad\qquad\qquad\qquad\qquad \text{ELSE} \\
\qquad\qquad\qquad\qquad\qquad\quad [delete \mapsto \text{TRUE}]] \\
\qquad\qquad\qquad \text{If all the change values are valid, record the changes required to roll} \\
\qquad\qquad\qquad \text{back the proposal and the index to which the rollback changes} \\
\qquad\qquad\qquad \text{will roll back the configuration.} \\
\qquad\qquad\quad \text{IN} \\
\qquad\qquad\qquad \lor proposal' = [proposal \text{ EXCEPT } ![i].rollback = [index\ \mapsto rollbackIndex, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad values \mapsto rollbackValues], \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].state\ \ = ProposalComplete] \\
\qquad\qquad\qquad \lor proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalFailed] \\
\qquad\qquad \text{For Rollback proposals, validate the rollback changes which are} \\
\qquad\qquad \text{proposal being rolled back.} \\
\qquad\quad \lor \land proposal[i].type = ProposalRollback \\
\qquad\qquad\quad \text{Rollbacks can only be performed on Change type proposals.} \\
\qquad\qquad \land \lor \land proposal[proposal[i].rollback.index].type = ProposalChange \\
\qquad\qquad\qquad\quad \text{Only roll back the change if it's the lastest change made}
\end{array}
$$

```
                to the configuration based on the configuration index.
            ∧ ∨ ∧ configuration.committed.index = proposal[i].rollback.index
                ∧ LET changeIndex  ≜ proposal[proposal[i].rollback.index].rollback.index
                      changeValues ≜ proposal[proposal[i].rollback.index].rollback.values
                      rollbackValues ≜ proposal[proposal[i].rollback.index].change.values
                    Record the changes required to roll back the target proposal and the index to
                    which the configuration is being rolled back.
                  IN  ∧ proposal' = [proposal EXCEPT ![i].change = [index  ↦ changeIndex,
                                                                    values ↦ changeValues],
                                                     ![i].change = [index  ↦ proposal[i].chang
                                                                    values ↦ changeValues],
                                                     ![i].state  = ProposalComplete]
                    If the Rollback target is not the most recent change to the configuration,
                    fail validation for the proposal.
                ∨ ∧ configuration.committed.index ≠ proposal[i].rollback.index
                  ∧ proposal' = [proposal EXCEPT ![i].state = ProposalFailed]
            If a Rollback proposal is attempting to roll back another Rollback,
            fail validation for the proposal.
          ∨ ∧ proposal[proposal[i].rollback.index].type = ProposalRollback
            ∧ proposal' = [proposal EXCEPT ![i].state = ProposalFailed]
        ∧ UNCHANGED ⟨configuration, target⟩
    ∨ ∧ proposal[i].state = ProposalComplete
      ∧ proposal' = [proposal EXCEPT ![i].phase = ProposalCommit,
                                     ![i].state = ProposalInProgress]
      ∧ UNCHANGED ⟨configuration, target⟩
While in the Commit state, commit the proposed changes to the configuration.
∨ ∧ proposal[i].phase = ProposalCommit
  ∧ ∨ ∧ proposal[i].state = ProposalInProgress
      Only commit the proposal if the prior proposal has already been committed.
      ∧ configuration.index = i − 1
      ∧ configuration' = [configuration EXCEPT !.committed.values = proposal[i].change.values,
                                               !.committed.index  = proposal[i].change.index,
                                               !.index            = i]
      ∧ proposal' = [proposal EXCEPT ![i].state = ProposalComplete]
      ∧ UNCHANGED ⟨target⟩
    ∨ ∧ proposal[i].state = ProposalComplete
      ∧ proposal' = [proposal EXCEPT ![i].phase = ProposalApply,
                                     ![i].state = ProposalInProgress]
      ∧ UNCHANGED ⟨configuration, target⟩
While in the Apply phase, apply the proposed changes to the target.
∨ ∧ proposal[i].phase = ProposalApply
  ∧ proposal[i].state  = ProposalInProgress
  ∧ configuration.applied.index = i − 1
  ∧ configuration.applied.term  = mastership.term
  ∧ mastership.master = n
```

Model successful and failed target update requests.

$\land \lor \land target' = [target \text{ EXCEPT } !.values = proposal[i].change.values]$
$\land configuration' = [configuration \text{ EXCEPT}$
$!.applied.index = i,$
$!.applied.values = proposal[i].change.values$
$@@ configuration.applied.values]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete]$

If the proposal could not be applied, update the configuration's applied index and mark the proposal Failed.

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

$\lor \land proposal[i].phase = ProposalAbort$
$\land proposal[i].state = ProposalInProgress$

The index will always be greater than or equal to the *applied.index*.
If only the index matches the previous proposal index, update
the index to enable commits of later proposals, but do not
mark the Abort phase Complete until the *applied.index* has been incremented.

$\land \lor \land configuration.index = i - 1$
$\land configuration' = [configuration \text{ EXCEPT } !.index = i]$
$\land \text{UNCHANGED } \langle proposal \rangle$

If the configuration's *applied.index* matches the previous proposal index,
update the *applied.index* and mark the proposal Complete for the Abort phase.

$\lor \land configuration.index \geq i$
$\land configuration.applied.index = i - 1$
$\land configuration' = [configuration \text{ EXCEPT } !.applied.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete]$

If both the configuration's index and *applied.index* match the
previous proposal index, update the index and *applied.index*
and mark the proposal Complete for the Abort phase.

$\lor \land configuration.index = i - 1$
$\land configuration.applied.index = i - 1$
$\land configuration' = [configuration \text{ EXCEPT } !.index = i,$
$!.applied.index = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete]$

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

$\land \text{UNCHANGED } \langle mastership, conn \rangle$

Formal specification, constraints, and theorems.

$InitProposal \triangleq$
$\land proposal = [$
$i \in \{\} \mapsto [$
$phase \mapsto ProposalInitialize,$

$state \mapsto ProposalInProgress]]$

$\land\ Trace!Init$

$NextProposal \triangleq$

$\lor \exists\, n \in Node :$

$\exists\, i \in \text{DOMAIN } proposal :$

$Trace!Step(ReconcileProposal(n,\ i),\ [node \mapsto n,\ index \mapsto i])$

* 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*