MODULE *Proposal*

EXTENDS *Configuration*, *Mastership*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

CONSTANT *NumProposals*

Transaction type constants
CONSTANTS
ProposalChange,
ProposalRollback

Phase constants
CONSTANTS
ProposalCommit,
ProposalApply

Status constants
CONSTANTS
ProposalPending,
ProposalInProgress,
ProposalComplete,
ProposalFailed

CONSTANT *TraceProposal*

A record of per-target proposals
VARIABLE *proposal*

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

LOCAL $NextState \triangleq [$
$proposals \mapsto [i \in \{i \in \text{DOMAIN } proposal' : proposal'[i].state \neq Nil\} \mapsto proposal'[i]]$,
$configuration \mapsto configuration'$,
$target \mapsto target'$,

$mastership \mapsto mastership'$,
$nodes \mapsto node']$

LOCAL $Trace \triangleq$ INSTANCE $Trace$ WITH
$Module \leftarrow$ "Proposal",
$InitState \leftarrow InitState$,
$NextState \leftarrow NextState$,
$Enabled \leftarrow TraceProposal$

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.
$\land configuration.commit.index = i - 1$
$\land configuration.commit.revision = configuration.commit.target$
$\land configuration' = [configuration$ EXCEPT $!.commit.index = i,$
$!.commit.target = i]$
$\land proposal' = [proposal$ 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.
$\land \lor$ LET $rollbackRevision \triangleq configuration.commit.revision$
$rollbackValues \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
IF $p \in$ DOMAIN $configuration.commit.values$ THEN
$configuration.commit.values[p]$
ELSE
$[index \mapsto 0,\ value \mapsto Nil]]$
$changeValues \triangleq [p \in$ DOMAIN $proposal[i].change.values \mapsto$
$proposal[i].change.values[p]$ @@ $[index \mapsto i]]$
IN $\land configuration' = [configuration$ EXCEPT $!.commit.revision = i,$
$!.commit.values = changeValues]$
$\land proposal' = [proposal$ EXCEPT $![i].change.values = changeValues,$
$![i].change.phase = ProposalApply,$
$![i].change.status = ProposalPending,$
$![i].rollback.revision = rollbackRevision,$
$![i].rollback.values = rollbackValues]$

$\lor \land configuration' = [configuration \text{ EXCEPT } !.commit.revision = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalFailed]$
$\land$ 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.

$ApplyChange(n,\ i) \triangleq$
$\land \lor \land proposal[i].change.status = ProposalPending$
To apply a change, the apply index must be the prior index.
$\land configuration.apply.index = i - 1$
$\land configuration.apply.revision = configuration.apply.target$
$\land configuration' = [configuration \text{ EXCEPT } !.apply.index = i,$
$!.apply.target = i]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalInProgress]$
$\land$ UNCHANGED $\langle target \rangle$
$\lor \land proposal[i].change.status = ProposalInProgress$
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.
$\land node[n].connected$
$\land mastership.conn = node[n].incarnation$
$\land target.running$
$\land node[n].target = target.incarnation$
Model successful and failed target update requests.
$\land \lor \land target' = [target \text{ EXCEPT } !.values = proposal[i].change.values \mathbin{@@} target.values]$
$\land$ LET $values \triangleq proposal[i].change.values \mathbin{@@} configuration.apply.values$
IN $configuration' = [configuration \text{ EXCEPT } !.apply.revision = i,$
$!.apply.incarnation = target.incarnation,$
$!.apply.values = values]$
$\land proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalComplete]$
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.
$\lor \land proposal' = [proposal \text{ EXCEPT } ![i].change.status = ProposalFailed]$
$\land$ 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.

$CommitRollback(n,\ i) \triangleq$
$\land \lor \land proposal[i].rollback.status = ProposalPending$
$\land configuration.commit.target = i$

```
        ∧ configuration.commit.revision = i
        ∧ configuration' = [configuration EXCEPT !.commit.target = proposal[i].rollback.revision]
        ∧ proposal' = [proposal EXCEPT ![i].rollback.status = ProposalInProgress]
     ∨ ∧ proposal[i].rollback.status = ProposalInProgress
        ∧ LET revision ≜ proposal[i].rollback.revision
              values   ≜ proposal[i].rollback.values
          IN  ∧ configuration' = [configuration EXCEPT !.commit.revision = revision,
                                                       !.commit.values   = values]
              ∧ proposal' = [proposal EXCEPT ![i].rollback.phase  = ProposalApply,
                                             ![i].rollback.status = ProposalPending]
  ∧ UNCHANGED ⟨target⟩
```

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) ≜
  ∧ ∨ ∧ proposal[i].rollback.status = ProposalPending
        ∧ configuration.apply.target = i
        ∧ configuration.apply.revision = i
        ∧ configuration' = [configuration EXCEPT !.apply.target = proposal[i].rollback.revision]
        ∧ proposal' = [proposal EXCEPT ![i].rollback.status = ProposalInProgress]
        ∧ UNCHANGED ⟨target⟩
    ∨ ∧ proposal[i].rollback.status = 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 revision ≜ proposal[i].rollback.revision
              values   ≜ proposal[i].rollback.values @@ configuration.apply.values
          IN  configuration' = [configuration EXCEPT !.apply.revision = revision,
                                                     !.apply.values   = values]
        ∧ proposal' = [proposal EXCEPT ![i].rollback.status = ProposalComplete]

ReconcileProposal(n, i) ≜
  ∧ mastership.master = n
  ∧ ∨ ∧ proposal[i].state = ProposalChange
        ∧ ∨ ∧ proposal[i].change.phase = ProposalCommit
            ∧ CommitChange(n, i)
          ∨ ∧ proposal[i].change.phase = ProposalApply
            ∧ ApplyChange(n, i)
    ∨ ∧ proposal[i].state = ProposalRollback
        ∧ ∨ ∧ proposal[i].rollback.phase = ProposalCommit
```

$\land$ $CommitRollback(n,\ i)$
$\lor$ $\land$ $proposal[i].rollback.phase = ProposalApply$
$\land$ $ApplyRollback(n,\ i)$
$\land$ UNCHANGED $\langle mastership,\ node \rangle$

Formal specification, constraints, and theorems.

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

$NextProposal \triangleq$
$\lor \exists\, n \in Nodes :$
$\exists\, i \in$ 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*