MODULE *Proposal*

EXTENDS *Configuration*, *Mastership*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

Transaction type constants
CONSTANTS
ProposalChange,
ProposalRollback

Phase constants
CONSTANTS
ProposalCommit,
ProposalApply

Status constants
CONSTANTS
ProposalInProgress,
ProposalComplete,
ProposalFailed

CONSTANT *TraceProposal*

A record of per-target proposals
VARIABLE *proposal*

LOCAL $InitState \triangleq [$
$proposals \mapsto proposal,$
$configuration \mapsto configuration,$
$target \mapsto target,$
$mastership \mapsto mastership,$
$nodes \mapsto node]$

LOCAL $NextState \triangleq [$
$proposals \mapsto proposal',$
$configuration \mapsto configuration',$
$target \mapsto target',$
$mastership \mapsto mastership',$
$nodes \mapsto node']$

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

$CommitChange(n,\ i) \triangleq$

Change proposals cannot be validated and committed until the prior index has been validated and committed and the configuration values updated.

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

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 $rollbackIndex \triangleq configuration.committed.revision$
$rollbackValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
IF $p \in \text{DOMAIN } configuration.committed.values$ THEN
$configuration.committed.values[p]$
ELSE
$[index \mapsto 0,\ value \mapsto Nil]]$
$changeValues \triangleq [p \in \text{DOMAIN } proposal[i].change.values \mapsto$
$proposal[i].change.values[p]$ @@ $[index \mapsto i]]$
IN $\land\ configuration' = [configuration$ EXCEPT $!.committed.index = i,$
$!.committed.revision = i,$
$!.committed.values = changeValues]$
$\land\ proposal' = [proposal$ EXCEPT $![i].change = [$
$index \mapsto i,$
$values \mapsto changeValues],$
$![i].rollback = [$
$index \mapsto rollbackIndex,$
$values \mapsto rollbackValues],$
$![i].phase = ProposalApply,$
$![i].state = ProposalInProgress]$

A proposal can fail validation at this point, in which case the proposal is marked failed.

$\lor\ \land configuration' = [configuration$ EXCEPT $!.committed.index = i]$
$\land\ proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$

$\land$ UNCHANGED $\langle target \rangle$

$CommitRollback(n,\ i) \triangleq$

Rollbacks can only be done on change type proposals.

$\land\ \lor\ \land\ proposal[proposal[i].rollback.index].type = ProposalChange$

If the change is the latest made to the configuration, roll it back.

$\land\ \lor\ \land\ configuration.committed.revision = proposal[i].rollback.index$

Record the changes required to roll back the target proposal and the index to

which the configuration is being rolled back.

$\land$ LET $changeIndex \triangleq proposal[proposal[i].rollback.index].rollback.index$
$changeValues \triangleq proposal[proposal[i].rollback.index].rollback.values$

Note: these two changes must be implemented as an atomic, idempotent update.
Implementations should check if the configuration has already been updated and
skip the configuration update if the committed index is $\geq$ the proposal index.

IN $\land configuration' = [configuration$ EXCEPT $!.committed.index = i,$
$!.committed.revision = changeIndex,$
$!.committed.values = changeValues]$

$\land proposal' = [proposal$ EXCEPT $![i].change = [$
$index \mapsto changeIndex,$
$values \mapsto changeValues],$
$![i].phase = ProposalApply,$
$![i].state = ProposalInProgress]$

If the change has not yet been committed to the configuration, abort it.

$\lor \land configuration.committed.revision < proposal[i].rollback.index$
$\land proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete,$
$![proposal[i].rollback.index].state = ProposalFailed]$
$\land$ UNCHANGED $\langle configuration \rangle$

If another change needs to be rolled back before this change can be,
fail the rollback.

$\lor \land configuration.committed.revision > proposal[i].rollback.index$
$\land proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$
$\land$ UNCHANGED $\langle configuration \rangle$

If a Rollback proposal is attempting to roll back another Rollback,
fail validation for the proposal.

$\lor \land proposal[proposal[i].rollback.index].type = ProposalRollback$
$\land proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$
$\land$ UNCHANGED $\langle configuration \rangle$

$\land$ UNCHANGED $\langle target \rangle$

$ApplyChange(n,\ i) \triangleq$

Change proposals cannot be applied until the prior index has
been applied and the configuration's applied index updated.

$\land configuration.applied.index = i - 1$

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

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

Verify the node's connection to the target.

$\land node[n].connected$
$\land target.running$

Model successful and failed target update requests.

$\land \lor \land target' = [target$ EXCEPT $!.values = proposal[i].change.values$ @@ $target.values]$
$\land$ LET $index \triangleq proposal[i].change.index$
$values \triangleq proposal[i].change.values$ @@ $configuration.applied.values$

IN $configuration' = [configuration$ EXCEPT $!.applied.index = i,$
$!.applied.revision = index,$
$!.applied.values = values]$

$\land\ proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$

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.

$\lor\ \land\ proposal' = [proposal$ EXCEPT $![i].state = ProposalFailed]$
$\land$ UNCHANGED $\langle configuration, target\rangle$

$ApplyRollback(n, i) \triangleq$

The target change cannot be rolled back until it has been applied or failed.

$\land\ configuration.applied.index = proposal[i].rollback.index$

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

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

Verify the node's connection to the target.

$\land\ node[n].connected$
$\land\ target.running$
$\land\ target' = [target$ EXCEPT $!.values = proposal[i].change.values$ @@ $target.values]$
$\land$ LET $index \triangleq proposal[i].change.index$
$values \triangleq proposal[i].change.values$ @@ $configuration.applied.values$
IN $configuration' = [configuration$ EXCEPT $!.applied.index = i,$
$!.applied.revision = index,$
$!.applied.values = values]$
$\land\ proposal' = [proposal$ EXCEPT $![i].state = ProposalComplete]$

$ReconcileProposal(n, i) \triangleq$

$\land\ mastership.master = n$
$\land\ \land\ \lor\ \land\ proposal[i].phase = ProposalCommit$
$\land\ proposal[i].state = ProposalInProgress$
$\land\ \lor\ \land\ proposal[i].type = ProposalChange$
$\land\ CommitChange(n, i)$
$\lor\ \land\ proposal[i].type = ProposalRollback$
$\land\ CommitRollback(n, i)$
$\lor\ \land\ proposal[i].phase = ProposalApply$
$\land\ proposal[i].state = ProposalInProgress$
$\land\ \lor\ \land\ proposal[i].type = ProposalChange$
$\land\ ApplyChange(n, i)$
$\lor\ \land\ proposal[i].type = ProposalRollback$
$\land\ ApplyRollback(n, i)$

If the proposal is complete, increment the applied index in sequential order to unblock proposals in the apply phase.

$\lor\ \land\ proposal[i].state \in \{ProposalComplete, ProposalFailed\}$
$\land\ configuration.applied.index = i - 1$

$\land$ $configuration' = [configuration$ EXCEPT $!.applied.index = i]$
$\land$ UNCHANGED $\langle proposal, target \rangle$
$\land$ UNCHANGED $\langle mastership, node \rangle$

Formal specification, constraints, and theorems.

$InitProposal \triangleq$
$\land$ $proposal = [$
$i \in \{\} \mapsto [$
$type \mapsto ProposalChange,$
$change \mapsto [$
$index \mapsto 0,$
$values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto Nil, delete \mapsto$ FALSE$]]],$
$rollback \mapsto [$
$index \mapsto 0,$
$values \mapsto [p \in \{\} \mapsto [index \mapsto 0, value \mapsto Nil, delete \mapsto$ FALSE$]]],$
$phase \mapsto ProposalCommit,$
$state \mapsto ProposalInProgress]]$
$\land$ $Trace!Init$

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