
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*,
 configurations \mapsto *configuration*,
 targets \mapsto *target*,
 masterships \mapsto *mastership*,
 nodes \mapsto *node*]

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

$\text{LOCAL } \text{Trace} \triangleq \text{INSTANCE } \text{Trace} \text{ WITH}$
 $\text{Module} \leftarrow \text{"Proposals"},$
 $\text{InitState} \leftarrow \text{InitState},$
 $\text{NextState} \leftarrow \text{NextState},$
 $\text{Enabled} \leftarrow \text{TraceProposal}$

Reconcile a proposal
 $\text{ReconcileProposal}(n, i) \triangleq$
 Only the master can process proposals for the target.
 $\wedge \text{mastership.master} = n$
 While in the Commit state, commit the proposed changes to the configuration.
 $\wedge \vee \wedge \text{proposal}[i].\text{phase} = \text{ProposalCommit}$
 $\wedge \vee \wedge \text{proposal}[i].\text{state} = \text{ProposalInProgress}$
 Only commit the proposal if the prior proposal has already been committed.
 $\wedge \text{configuration.committed.index} = i - 1$
 For Change proposals validate the set of requested changes.
 $\wedge \vee \wedge \text{proposal}[i].\text{type} = \text{ProposalChange}$
 If all the change values are valid, record the changes required to roll
 back the proposal and the revision to which the rollback changes
 will roll back the configuration.
 $\wedge \vee \text{LET } \text{rollbackRevision} \triangleq \text{configuration.committed.revision}$
 $\text{rollbackValues} \triangleq [p \in \text{DOMAIN } \text{proposal}[i].\text{change.values} \mapsto$
 $\text{IF } p \in \text{DOMAIN } \text{configuration.committed.values} \text{ THEN}$
 $\text{configuration.committed.values}[p]$
 ELSE
 $[\text{delete} \mapsto \text{TRUE}]]$
 $\text{changeValues} \triangleq [p \in \text{DOMAIN } \text{proposal}[i].\text{change.values} \mapsto$
 $\text{proposal}[i].\text{change.values}[p] \text{ @@ } [\text{index} \mapsto i]]$
 $\text{IN } \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.revision} = i,$
 $!\text{committed.values} = \text{changeValues}]$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{change} = [$
 $\text{revision} \mapsto i,$
 $\text{values} \mapsto \text{changeValues}],$
 $![i].\text{rollback} = [$
 $\text{revision} \mapsto \text{rollbackRevision},$
 $\text{values} \mapsto \text{rollbackValues}],$
 $![i].\text{state} = \text{ProposalComplete}]$
 A proposal can fail validation at this point, in which case the proposal
 is marked failed.
 $\vee \wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{state} = \text{ProposalFailed}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration} \rangle$
 For Rollback proposals, validate the rollback changes which are
 proposal being rolled back.

$\vee \wedge \text{proposal}[i].\text{type} = \text{ProposalRollback}$
 Rollbacks can only be performed on Change type proposals.
 $\wedge \vee \wedge \text{proposal}[\text{proposal}[i].\text{rollback.index}].\text{type} = \text{ProposalChange}$
 Only roll back the change if it's the latest change made
 to the configuration based on the configuration revision.
 $\wedge \vee \wedge \text{configuration.committed.revision} = \text{proposal}[i].\text{rollback.index}$
 Record the changes required to roll back the target proposal and the index to
 which the configuration is being rolled back.
 $\wedge \text{LET } \text{changeRevision} \triangleq \text{proposal}[\text{proposal}[i].\text{rollback.index}].\text{rollback.revision}$
 $\text{changeValues} \triangleq \text{proposal}[\text{proposal}[i].\text{rollback.index}].\text{rollback.values}$
 IN $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.revision} = \text{changeRevision}$
 $!.\text{committed.values} = \text{changeValues}]$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{change} = [$
 $\text{revision} \mapsto \text{changeRevision},$
 $\text{values} \mapsto \text{changeValues}],$
 $![i].\text{state} = \text{ProposalComplete}]$
 If the Rollback target is not the most recent change to the configuration,
 fail validation for the proposal.
 $\vee \wedge \text{configuration.committed.revision} \neq \text{proposal}[i].\text{rollback.index}$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{state} = \text{ProposalFailed}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration} \rangle$
 If a Rollback proposal is attempting to roll back another Rollback,
 fail validation for the proposal.
 $\vee \wedge \text{proposal}[\text{proposal}[i].\text{rollback.index}].\text{type} = \text{ProposalRollback}$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{state} = \text{ProposalFailed}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{target} \rangle$
 Once the proposal is committed, update the configuration's commit index
 and move to the apply phase.
 $\vee \wedge \text{proposal}[i].\text{state} = \text{ProposalComplete}$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.index} = i]$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{phase} = \text{ProposalApply},$
 $![i].\text{state} = \text{ProposalInProgress}]$
 $\wedge \text{UNCHANGED } \langle \text{target} \rangle$
 If the proposal fails, mark the configuration applied for the proposal index.
 $\vee \wedge \text{proposal}[i].\text{state} = \text{ProposalFailed}$
 $\wedge \vee \wedge \text{configuration.committed.index} = i - 1$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.index} = i]$
 $\vee \wedge \text{configuration.committed.index} \geq i$
 $\wedge \text{configuration.applied.index} = i - 1$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{applied.index} = i]$
 $\wedge \text{UNCHANGED } \langle \text{proposal}, \text{target} \rangle$
 While in the Apply phase, apply the proposed changes to the target.
 $\vee \wedge \text{proposal}[i].\text{phase} = \text{ProposalApply}$
 For the proposal to be applied, the node must be connected to a running target.

$$\begin{aligned}
& \text{revision} \mapsto 0, \\
& \text{values} \mapsto [p \in \{\} \mapsto [\text{index} \mapsto 0, \text{value} \mapsto \text{Nil}, \text{delete} \mapsto \text{FALSE}]]], \\
& \text{phase} \mapsto \text{ProposalCommit}, \\
& \text{state} \mapsto \text{ProposalInProgress}] \\
& \wedge \text{Trace!Init}
\end{aligned}$$

$$\begin{aligned}
\text{NextProposal} & \triangleq \\
& \vee \exists n \in \text{Node} : \\
& \quad \exists i \in \text{DOMAIN } \text{proposal} : \\
& \quad \text{Trace!Step}(\text{ReconcileProposal}(n, i), [\text{node} \mapsto n, \text{index} \mapsto i])
\end{aligned}$$

* 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*