
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'*]

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

$\text{CommitChange}(n, i) \triangleq$

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

$\wedge \text{configuration.committed.index} = i - 1$

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.

$\wedge \vee \text{LET } \text{rollbackIndex} \triangleq \text{configuration.committed.revision}$

$\text{rollbackValues} \triangleq [p \in \text{DOMAIN } \text{proposal}[i].\text{change.values} \mapsto$

$\text{IF } p \in \text{DOMAIN } \text{configuration.committed.values} \text{ THEN}$
 $\text{configuration.committed.values}[p]$

ELSE

$[index \mapsto 0, \text{value} \mapsto \text{Nil}]$

$\text{changeValues} \triangleq [p \in \text{DOMAIN } \text{proposal}[i].\text{change.values} \mapsto$

$\text{proposal}[i].\text{change.values}[p] \text{ @@ } [index \mapsto i]]$

IN $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.index} = i,$
 $!.committed.revision = i,$
 $!.committed.values = \text{changeValues}]$

$\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{change} = [$

$index \mapsto i,$

$values \mapsto \text{changeValues}],$

$![i].\text{rollback} = [$

$index \mapsto \text{rollbackIndex},$

$values \mapsto \text{rollbackValues}],$

$![i].\text{phase} = \text{ProposalApply},$

$![i].\text{state} = \text{ProposalInProgress}]$

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

$\vee \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.index} = i]$

$\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{state} = \text{ProposalFailed}]$

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

$\text{CommitRollback}(n, i) \triangleq$

Rollbacks can only be done on change type proposals.

$\wedge \vee \wedge \text{proposal}[\text{proposal}[i].\text{rollback.index}].\text{type} = \text{ProposalChange}$

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

$\wedge \vee \wedge \text{configuration.committed.revision} = \text{proposal}[i].\text{rollback.index}$

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

which the configuration is being rolled back.

\wedge 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 $\wedge configuration' = [configuration \text{ EXCEPT } !.committed.index = i,$
 $!.committed.revision = changeIndex,$
 $!.committed.values = changeValues]$
 $\wedge proposal' = [proposal \text{ EXCEPT } ![i].change = [$
 $index \mapsto changeIndex,$
 $values \mapsto changeValues],$
 $![i].phase = ProposalApply,$
 $![i].state = ProposalInProgress]$

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

$\vee \wedge configuration.committed.revision < proposal[i].rollback.index$
 $\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete,$
 $![proposal[i].rollback.index].state = ProposalFailed]$
 $\wedge \text{UNCHANGED } \langle configuration \rangle$

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

$\vee \wedge configuration.committed.revision > proposal[i].rollback.index$
 $\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalFailed]$
 $\wedge \text{UNCHANGED } \langle configuration \rangle$

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

$\vee \wedge proposal[proposal[i].rollback.index].type = ProposalRollback$
 $\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalFailed]$
 $\wedge \text{UNCHANGED } \langle configuration \rangle$

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

$ApplyChange(n, i) \triangleq$

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

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

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

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

Verify the node's connection to the target.

$\wedge node[n].connected$

$\wedge target.running$

Model successful and failed target update requests.

$\wedge \vee \wedge target' = [target \text{ EXCEPT } !.values = proposal[i].change.values]$
 \wedge LET $index \triangleq proposal[i].change.index$
 $values \triangleq proposal[i].change.values @@@ configuration.applied.values$

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

$\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete]$

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.

$\vee \wedge configuration' = [configuration \text{ EXCEPT } !.applied.index = i]$

$\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalFailed]$

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

$ApplyRollback(n, i) \triangleq$

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

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

Verify the node's connection to the target.

$\wedge node[n].connected$

$\wedge target.running$

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

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

$\wedge target' = [target \text{ EXCEPT } !.values = proposal[i].change.values]$

$\wedge \text{LET } index \triangleq proposal[i].change.index$

$values \triangleq proposal[i].change.values @@ configuration.applied.values$

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

$\wedge proposal' = [proposal \text{ EXCEPT } ![i].state = ProposalComplete]$

$ReconcileProposal(n, i) \triangleq$

$\wedge mastership.master = n$

$\wedge \vee \wedge proposal[i].phase = ProposalCommit$

$\wedge proposal[i].state = ProposalInProgress$

$\wedge \vee \wedge proposal[i].type = ProposalChange$

$\wedge CommitChange(n, i)$

$\vee \wedge proposal[i].type = ProposalRollback$

$\wedge CommitRollback(n, i)$

$\vee \wedge proposal[i].phase = ProposalApply$

$\wedge proposal[i].state = ProposalInProgress$

$\wedge \vee \wedge proposal[i].type = ProposalChange$

$\wedge ApplyChange(n, i)$

$\vee \wedge proposal[i].type = ProposalRollback$

$\wedge ApplyRollback(n, i)$

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

$\vee \wedge proposal[i].state \in \{ProposalComplete, ProposalFailed\}$

$$\begin{aligned}
& \wedge \text{configuration.applied.index} = i - 1 \\
& \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{applied.index} = i] \\
& \wedge \text{UNCHANGED } \langle \text{proposal}, \text{target} \rangle \\
& \wedge \text{UNCHANGED } \langle \text{mastership}, \text{node} \rangle
\end{aligned}$$

Formal specification, constraints, and theorems.

$$\begin{aligned}
\text{InitProposal} & \triangleq \\
& \wedge \text{proposal} = [\\
& \quad i \in \{\} \mapsto [\\
& \quad \quad \text{type} \mapsto \text{ProposalChange}, \\
& \quad \quad \text{change} \mapsto [\\
& \quad \quad \quad \text{index} \mapsto 0, \\
& \quad \quad \quad \text{values} \mapsto [p \in \{\} \mapsto [\text{index} \mapsto 0, \text{value} \mapsto \text{Nil}, \text{delete} \mapsto \text{FALSE}]]], \\
& \quad \quad \text{rollback} \mapsto [\\
& \quad \quad \quad \text{index} \mapsto 0, \\
& \quad \quad \quad \text{values} \mapsto [p \in \{\} \mapsto [\text{index} \mapsto 0, \text{value} \mapsto \text{Nil}, \text{delete} \mapsto \text{FALSE}]]], \\
& \quad \quad \text{phase} \mapsto \text{ProposalCommit}, \\
& \quad \quad \text{state} \mapsto \text{ProposalInProgress}] \\
& \wedge \text{Trace!Init} \\
\\
\text{NextProposal} & \triangleq \\
& \vee \exists n \in \text{Nodes} : \\
& \quad \exists i \in \text{DOMAIN } \text{proposal} : \\
& \quad \quad \text{Trace!Step}(\text{ReconcileProposal}(n, i), [\text{node} \mapsto n, \text{index} \mapsto i])
\end{aligned}$$

* 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*