
MODULE *Proposal*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant
CONSTANT *Nil*

Event constants
CONSTANTS
 Change,
 Rollback

Phase constants
CONSTANTS
 Commit,
 Apply

$Phase \triangleq$
 {*Nil*,
 Commit,
 Apply}

Status constants
CONSTANTS
 Pending,
 InProgress,
 Complete,
 Failed

$Status \triangleq$
 {*Nil*,
 Pending,
 InProgress,
 Complete,
 Failed}

The set of all nodes
CONSTANT *Node*

Variables defined by other modules.

VARIABLES

configuration,
mastership,
target

A record of per-target proposals

VARIABLE *proposal*

A sequence of configuration changes used for model checking.

VARIABLE *history*

TypeOK \triangleq

$\forall i \in \text{DOMAIN } \textit{proposal} :$
 $\wedge \textit{proposal}[i].\textit{change.phase} \in \textit{Phase}$
 $\wedge \textit{proposal}[i].\textit{change.state} \in \textit{Status}$
 $\wedge \forall p \in \text{DOMAIN } \textit{proposal}[i].\textit{change.values} :$
 $\wedge \textit{proposal}[i].\textit{change.values}[p].\textit{index} \in \textit{Nat}$
 $\wedge \textit{proposal}[i].\textit{change.values}[p].\textit{value} \neq \textit{Nil} \Rightarrow$
 $\quad \textit{proposal}[i].\textit{change.values}[p].\textit{value} \in \textit{STRING}$
 $\wedge \textit{proposal}[i].\textit{rollback.phase} \in \textit{Phase}$
 $\wedge \textit{proposal}[i].\textit{rollback.state} \in \textit{Status}$
 $\wedge \textit{proposal}[i].\textit{rollback.revision} \in \textit{Nat}$
 $\wedge \forall p \in \text{DOMAIN } \textit{proposal}[i].\textit{rollback.values} :$
 $\wedge \textit{proposal}[i].\textit{rollback.values}[p].\textit{index} \in \textit{Nat}$
 $\wedge \textit{proposal}[i].\textit{rollback.values}[p].\textit{value} \neq \textit{Nil} \Rightarrow$
 $\quad \textit{proposal}[i].\textit{rollback.values}[p].\textit{value} \in \textit{STRING}$

Test \triangleq INSTANCE *Test* WITH

File \leftarrow "Proposal.log",
CurrState \leftarrow [
 $\textit{proposals} \mapsto \textit{proposal}$,
 $\textit{configuration} \mapsto \textit{configuration}$,
 $\textit{mastership} \mapsto \textit{mastership}$,
 $\textit{target} \mapsto \textit{target}$],
SuccState \leftarrow [
 $\textit{proposals} \mapsto \textit{proposal}'$,
 $\textit{configuration} \mapsto \textit{configuration}'$,
 $\textit{mastership} \mapsto \textit{mastership}'$,
 $\textit{target} \mapsto \textit{target}'$]

LOCAL *Max*(*s*) \triangleq CHOOSE $i \in s : \forall j \in s : i > j$

CommitChange(*n*, *i*) \triangleq
 $\wedge \textit{proposal}[i].\textit{change.phase} = \textit{Commit}$

$\wedge \text{proposal}[i].\text{change.state} = \text{InProgress}$
 If the committed index does not match the proposal index, commit the change.
 $\wedge \vee \wedge \text{configuration.committed.index} = i - 1$
 If the change is valid, update the committed index, revision, and values.
 $\wedge \vee \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } \begin{array}{l} \text{!.committed.index} = i, \\ \text{!.committed.revision} = i, \\ \text{!.committed.values} = \text{proposal}[i].\text{change.values} @@ \\ \text{configuration.committed.values} \end{array}$
 $\wedge \text{history}' = \text{Append}(\text{history}, [\text{type} \mapsto \text{Change}, \text{phase} \mapsto \text{Commit}, \text{index} \mapsto i])$
 If the change is invalid, update only the committed index.
 $\vee \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } \text{!.committed.index} = i]$
 $\wedge \text{UNCHANGED } \langle \text{history} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$
 If both the committed index and committed revision were updated, the proposal was successful.
 $\vee \wedge \text{configuration.committed.index} = i$
 $\wedge \text{configuration.committed.revision} = i$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } \text{!.change.state} = \text{Complete}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{history} \rangle$
 If the committed index was updated but the revision was not, the proposal failed validation.
 $\vee \wedge \text{configuration.committed.index} = i$
 $\wedge \text{configuration.committed.revision} \neq i$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } \text{!.change.state} = \text{Failed}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{history} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{target} \rangle$

$\text{ApplyChange}(n, i) \triangleq$
 $\wedge \text{proposal}[i].\text{change.phase} = \text{Apply}$
 $\wedge \text{proposal}[i].\text{change.state} = \text{InProgress}$
 If the applied index does not match the proposal index, apply the change.
 $\wedge \vee \wedge \text{configuration.applied.index} = i - 1$
 $\wedge \text{configuration.state} = \text{Complete}$
 $\wedge \text{configuration.term} = \text{mastership.term}$
 If the change can be applied, update the index, revision, and values.
 $\wedge \vee \wedge \text{target}' = [\text{target} \text{ EXCEPT } \text{!.values} = \text{proposal}[i].\text{change.values} @@ \text{target.values}]$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } \begin{array}{l} \text{!.applied.index} = i, \\ \text{!.applied.revision} = i, \\ \text{!.applied.values} = \text{proposal}[i].\text{change.values} @@ \\ \text{configuration.applied.values} \end{array}$
 $\wedge \text{history}' = \text{Append}(\text{history}, [\text{type} \mapsto \text{Change}, \text{phase} \mapsto \text{Apply}, \text{index} \mapsto i])$
 If the change is invalid, update only the applied index.
 $\vee \wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } \text{!.applied.index} = i]$
 $\wedge \text{UNCHANGED } \langle \text{target}, \text{history} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$
 If the applied index and revision both match the proposal index, the change was successful.
 $\vee \wedge \text{configuration.applied.index} = i$

$\wedge \text{configuration.applied.revision} = i$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{change.state} = \text{Complete}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{target}, \text{history} \rangle$
 If the applied index matches the proposal index but the revision does not, the proposal failed.
 $\vee \wedge \text{configuration.applied.index} = i$
 $\wedge \text{configuration.applied.revision} \neq i$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{change.state} = \text{Failed}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{target}, \text{history} \rangle$

$\text{CommitRollback}(n, i) \triangleq$
 $\wedge \text{proposal}[i].\text{rollback.phase} = \text{Commit}$
 $\wedge \text{proposal}[i].\text{rollback.state} = \text{InProgress}$
 If the committed revision matches the proposal revision, roll back to the previous revision.
 $\wedge \vee \wedge \text{configuration.committed.revision} = i$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{committed.revision} = \text{proposal}[i].\text{rollback.revision},$
 $\quad \quad \quad !.\text{committed.values} = \text{proposal}[i].\text{rollback.values} @@$
 $\quad \quad \quad \text{configuration.committed.values}]$
 $\wedge \text{history}' = \text{Append}(\text{history}, [\text{type} \mapsto \text{Rollback}, \text{phase} \mapsto \text{Commit}, \text{index} \mapsto i])$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$
 If the committed index matches the rollback index, complete the rollback.
 $\vee \wedge \text{configuration.committed.revision} = \text{proposal}[i].\text{rollback.revision}$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{rollback.state} = \text{Complete}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{history} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{target} \rangle$

$\text{ApplyRollback}(n, i) \triangleq$
 $\wedge \text{proposal}[i].\text{rollback.phase} = \text{Apply}$
 $\wedge \text{proposal}[i].\text{rollback.phase} = \text{InProgress}$
 If the applied revision matches the proposal revision, roll back to the previous revision.
 $\wedge \vee \wedge \text{configuration.applied.revision} = i$
 $\wedge \text{target}' = [\text{target} \text{ EXCEPT } !.\text{values} = \text{proposal}[i].\text{rollback.values} @@ \text{target.values}]$
 $\wedge \text{configuration}' = [\text{configuration} \text{ EXCEPT } !.\text{applied.revision} = \text{proposal}[i].\text{rollback.revision},$
 $\quad \quad \quad !.\text{applied.values} = \text{proposal}[i].\text{rollback.values} @@$
 $\quad \quad \quad \text{configuration.applied.values}]$
 $\wedge \text{history}' = \text{Append}(\text{history}, [\text{type} \mapsto \text{Rollback}, \text{phase} \mapsto \text{Apply}, \text{index} \mapsto i])$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$
 If the committed index matches the rollback index, complete the rollback.
 $\vee \wedge \text{configuration.committed.revision} = \text{proposal}[i].\text{rollback.revision}$
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![i].\text{rollback.state} = \text{Complete}]$
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{target}, \text{history} \rangle$

Reconcile a proposal
 $\text{ReconcileProposal}(n, i) \triangleq$
 $\wedge i \in \text{DOMAIN } \text{proposal}$
 $\wedge \text{mastership.master} = n$
 $\wedge \vee \text{CommitChange}(n, i)$

$$\begin{aligned}
& \vee \textit{ApplyChange}(n, i) \\
& \vee \textit{CommitRollback}(n, i) \\
& \vee \textit{ApplyRollback}(n, i) \\
& \wedge \text{UNCHANGED } \langle \textit{mastership} \rangle
\end{aligned}$$
