MODULE *Config*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

An empty constant

CONSTANT *Nil*

Transaction type constants

CONSTANTS
TransactionChange,
TransactionRollback

Transaction isolation constants

CONSTANTS
IsolationDefault,
IsolationSerializable

Transaction status constants

CONSTANTS
TransactionInitializing,
TransactionInitialized,
TransactionValidating,
TransactionValidated,
TransactionCommitting,
TransactionCommitted,
TransactionApplying,
TransactionApplied,
TransactionFailed

$$\begin{aligned}&\mathit{TransactionStatus} \triangleq \\ &\quad \langle \mathit{TransactionInitializing}, \\ &\quad\ \mathit{TransactionInitialized}, \\ &\quad\ \mathit{TransactionValidating}, \\ &\quad\ \mathit{TransactionValidated}, \\ &\quad\ \mathit{TransactionCommitting}, \\ &\quad\ \mathit{TransactionCommitted}, \\ &\quad\ \mathit{TransactionApplying}, \\ &\quad\ \mathit{TransactionApplied}, \\ &\quad\ \mathit{TransactionFailed}\rangle\end{aligned}$$

Proposal type constants

CONSTANTS
ProposalChange,
ProposalRollback

Proposal status constants

CONSTANTS
ProposalInitializing,
ProposalInitialized,
ProposalValidating,
ProposalValidated,
ProposalCommitting,
ProposalCommitted,
ProposalApplying,
ProposalApplied,
ProposalFailed

$ProposalStatus \triangleq$
$\langle ProposalInitializing,$
$ProposalInitialized,$
$ProposalValidating,$
$ProposalValidated,$
$ProposalCommitting,$
$ProposalCommitted,$
$ProposalApplying,$
$ProposalApplied,$
$ProposalFailed\rangle$

Configuration status constants

CONSTANTS
ConfigurationUnknown,
ConfigurationSynchronizing,
ConfigurationSynchronized,
ConfigurationPersisted,
ConfigurationFailed

CONSTANTS
Valid,
Invalid

CONSTANTS
Success,
Failure

The set of all nodes

CONSTANT *Node*

Target is the set of all targets and their possible paths and values.

Example: $Target \triangleq$ [
$target1 \mapsto$ [$persistent \mapsto$ FALSE, $values \mapsto$ [
$path1 \mapsto$ {"$value1$", "$value2$"},
$path2 \mapsto$ {"$value2$", "$value3$"}]],
$target2 \mapsto$ [$persistent \mapsto$ TRUE, $values \mapsto$ [
$path2 \mapsto$ {"$value3$", "$value4$"},
$path3 \mapsto$ {"$value4$", "$value5$"}]]]

CONSTANT $Target$

$Phase(S, s) \triangleq$ CHOOSE $i \in$ DOMAIN $S : S[i] = s$

$TransactionPhase(s) \triangleq Phase(TransactionStatus, s)$

$ProposalPhase(s) \triangleq Phase(ProposalStatus, s)$

ASSUME $Nil \in$ STRING

ASSUME $TransactionInitializing \in$ STRING
ASSUME $TransactionInitialized \in$ STRING
ASSUME $TransactionValidating \in$ STRING
ASSUME $TransactionValidated \in$ STRING
ASSUME $TransactionCommitting \in$ STRING
ASSUME $TransactionCommitted \in$ STRING
ASSUME $TransactionApplying \in$ STRING
ASSUME $TransactionApplied \in$ STRING
ASSUME $TransactionFailed \in$ STRING

ASSUME $ProposalInitializing \in$ STRING
ASSUME $ProposalInitialized \in$ STRING
ASSUME $ProposalValidating \in$ STRING
ASSUME $ProposalValidated \in$ STRING
ASSUME $ProposalCommitting \in$ STRING
ASSUME $ProposalCommitted \in$ STRING
ASSUME $ProposalApplying \in$ STRING
ASSUME $ProposalApplied \in$ STRING
ASSUME $ProposalFailed \in$ STRING

ASSUME $ConfigurationUnknown \in$ STRING
ASSUME $ConfigurationSynchronizing \in$ STRING
ASSUME $ConfigurationSynchronized \in$ STRING
ASSUME $ConfigurationPersisted \in$ STRING
ASSUME $ConfigurationFailed \in$ STRING

ASSUME $\land IsFiniteSet(Node)$
$\land \forall n \in Node :$
$\land n \notin$ DOMAIN $Target$
$\land n \in$ STRING

ASSUME $\land \forall\, t \in \text{DOMAIN } Target :$
$\land\ t \notin Node$
$\land\ t \in \text{STRING}$
$\land\ Target[t].persistent \in \text{BOOLEAN}$
$\land \forall\, p \in \text{DOMAIN } Target[t].values :$
$IsFiniteSet(\, Target[t].values[p])$

Configuration update/rollback requests are tracked and processed through two data types. Transactions represent the lifecycle of a single configuration change request and are stored in an append-only log. Configurations represent the desired configuration of a *gNMI* target based on the aggregate of relevant changes in the Transaction log.

TYPE $TransactionType ::= type \in$
$\{TransactionChange,$
$TransactionRollback\}$

TYPE $TransactionStatus ::= status \in$
$\{TransactionInitializing,$
$TransactionInitialized,$
$TransactionValidating,$
$TransactionValidated,$
$TransactionCommitting,$
$TransactionCommitted,$
$TransactionApplying,$
$TransactionApplied,$
$TransactionFailed\}$

TYPE Transaction $\triangleq$ [
$type ::= type \in TransactionType,$
$index ::= index \in Nat,$
$isolation ::= isolation \in \{IsolationDefault, IsolationSerializable\}$
$values ::=$ [
$target \in \text{SUBSET } (\text{DOMAIN } Target) \mapsto [\, path \in \text{SUBSET } (\text{DOMAIN } Target[target].values) \mapsto$
[
$value ::= value \in \text{STRING},$
$delete ::= delete \in \text{BOOLEAN}\ ]]],$
$rollback ::= index \in Nat,$
$targets ::= targets \in \text{SUBSET } (\text{DOMAIN } Target)$
$status ::= status \in TransactionStatus]$

TYPE $ProposalStatus ::= status \in$
$\{ProposalInitializing,$
$ProposalInitialized,$
$ProposalValidating,$
$ProposalValidated,$
$ProposalCommitting,$
$ProposalCommitted,$
$ProposalApplying,$
$ProposalApplied,$
$ProposalFailed\}$

```
TYPE Proposal ≜ [
  index          ::= index ∈ Nat,
  values         ::= [ path ∈ SUBSET (DOMAIN Target[target].values) ↦ [
      value ::= value ∈ STRING,
      delete ::= delete ∈ BOOLEAN ]],
  rollback       ::= index ∈ Nat,
  prevIndex      ::= prevIndex ∈ Nat,
  nextIndex      ::= nextIndex ∈ Nat,
  rollbackIndex ::= rollbackIndex ∈ Nat,
  rollbackValues ::= [ path ∈ SUBSET (DOMAIN Target[target].values) ↦ [
      value ::= value ∈ STRING,
      delete ::= delete ∈ BOOLEAN ]],
  status         ::= status ∈ ProposalStatus]

TYPE ConfigurationStatus ::= status ∈
  {ConfigurationUnknown,
   ConfigurationSynchronizing,
   ConfigurationSynchronized,
   ConfigurationPersisted,
   ConfigurationFailed}

TYPE Configuration ≜ [
  id             ::= id ∈ STRING,
  target         ::= target ∈ STRING,
  values         ::= [ path ∈ SUBSET (DOMAIN Target[target]) ↦ [
      value ::= value ∈ STRING,
      index ::= index ∈ Nat,
      deleted ::= delete ∈ BOOLEAN ]],
  configIndex ::= configIndex ∈ Nat,
  configTerm     ::= configTerm ∈ Nat,
  proposedIndex ::= proposedIndex ∈ Nat,
  committedIndex ::= committedIndex ∈ Nat,
  appliedIndex ::= appliedIndex ∈ Nat,
  appliedTerm ::= appliedTerm ∈ Nat,
  appliedValues ::= [ path ∈ SUBSET (DOMAIN Target[target]) ↦ [
      value ::= value ∈ STRING,
      index ::= index ∈ Nat,
      deleted ::= delete ∈ BOOLEAN ]],
  status ::= status ∈ ConfigurationStatus]
```

A transaction log. Transactions may either request a set
of changes to a set of targets or rollback a prior change.

VARIABLE *transaction*

A record of per-target proposals

VARIABLE *proposal*

A record of per-target configurations

VARIABLE *configuration*

A record of target states

VARIABLE *target*

A record of target masterships

VARIABLE *mastership*

$vars \triangleq \langle transaction,\ proposal,\ configuration,\ mastership,\ target \rangle$

This section models *mastership* for the configuration service.

Mastership is used primarily to track the lifecycle of individual configuration targets and react to state changes on the southbound. Each target is assigned a master from the *Node* set, and masters can be unset when the target disconnects.

Set node n as the master for target t

$SetMaster(n,\ t) \triangleq$
$\quad \land\ mastership[t].master \neq n$
$\quad \land\ mastership' = [mastership\ \text{EXCEPT}\ ![t].term\ \ = mastership[t].term + 1,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![t].master = n]$
$\quad \land\ \text{UNCHANGED}\ \langle transaction,\ proposal,\ configuration,\ target \rangle$

$UnsetMaster(t) \triangleq$
$\quad \land\ mastership[t].master \neq Nil$
$\quad \land\ mastership' = [mastership\ \text{EXCEPT}\ ![t].master = Nil]$
$\quad \land\ \text{UNCHANGED}\ \langle transaction,\ proposal,\ configuration,\ target \rangle$

This section models configuration changes and rollbacks. Changes are appended to the transaction log and processed asynchronously.

$Value(s,\ t,\ p) \triangleq$
$\quad \text{LET}\ value \triangleq \text{CHOOSE}\ v \in s : v.target = t \land v.path = p$
$\quad \text{IN}$
$\qquad [value\ \mapsto value.value,$
$\qquad\ delete \mapsto value.delete]$

$Paths(s,\ t) \triangleq$
$\quad [p \in \{v.path : v \in \{v \in s : v.target = t\}\} \mapsto Value(s,\ t,\ p)]$

$Changes(s) \triangleq$
$\quad [t \in \{v.target : v \in s\} \mapsto Paths(s,\ t)]$

$ValidValues(t,\ p) \triangleq$
$\quad \text{UNION}\ \{\{[value \mapsto v,\ delete \mapsto \text{FALSE}] : v \in Target[t].values[p]\},\ \{[value \mapsto Nil,\ delete \mapsto \text{TRUE}]\}\}$

$ValidPaths(t) \triangleq$
$\quad \text{UNION}\ \{\{v\ @@\ [path \mapsto p] : v \in ValidValues(t,\ p)\} : p \in \text{DOMAIN}\ Target[t].values\}$

$ValidTargets \triangleq$
$\quad \text{UNION}\ \{\{p\ @@\ [target \mapsto t] : p \in ValidPaths(t)\} : t \in \text{DOMAIN}\ Target\}$

```
The set of all valid sets of changes to all targets and their paths.
The set of possible changes is computed from the Target model value.
ValidChanges ≜
  LET changeSets ≜ {s ∈ SUBSET ValidTargets :
                      ∀ t ∈ DOMAIN Target :
                        ∀ p ∈ DOMAIN Target[t].values :
                          Cardinality({v ∈ s : v.target = t ∧ v.path = p}) ≤ 1}
  IN
    {Changes(s) : s ∈ changeSets}

The next available index in the transaction log.
This is computed as the max of the existing indexes in the log to
allow for changes to the log (e.g. log compaction) to be modeled.
NextIndex ≜
  IF DOMAIN transaction = {} THEN
    1
  ELSE
    LET i ≜ CHOOSE i ∈ DOMAIN transaction :
          ∀ j ∈ DOMAIN transaction : i ≥ j
    IN  i + 1

Add a set of changes 'c' to the transaction log
Change(c) ≜
  ∧  ∃ isolation ∈ {IsolationDefault, IsolationSerializable} :
       ∧ transaction' = transaction @@ (NextIndex :> [type      ↦ TransactionChange,
                                                      index     ↦ NextIndex,
                                                      isolation ↦ isolation,
                                                      values    ↦ c,
                                                      targets   ↦ {},
                                                      status    ↦ TransactionInitializing])
  ∧  UNCHANGED ⟨proposal, configuration, mastership, target⟩

Add a rollback of transaction 't' to the transaction log
Rollback(t) ≜
  ∧ ∃ isolation ∈ {IsolationDefault, IsolationSerializable} :
      ∧ transaction' = transaction @@ (NextIndex :> [type      ↦ TransactionRollback,
                                                     index     ↦ NextIndex,
                                                     isolation ↦ isolation,
                                                     rollback  ↦ t,
                                                     targets   ↦ {},
                                                     status    ↦ TransactionInitializing])
  ∧ UNCHANGED ⟨proposal, configuration, mastership, target⟩
```

This section models the Transaction log reconciler.

Transactions come in two flavors : − *Change* transactions contain a set of changes to be applied to a set of *targets* − *Rollback* transactions reference a prior change transaction to be reverted to the previous state

Both types of transaction are reconciled in stages:
* Pending - waiting for prior transactions to complete
* Validating - validating the requested changes
* Applying - applying the changes to target configurations
* Complete - completed applying changes successfully
* Failed - failed applying changes

Reconcile a transaction

$$
\begin{array}{l}
ReconcileTransaction(n,\ i)\ \triangleq \\
\quad \land\ \lor\ \land\ transaction[i].status = TransactionInitializing \\
\qquad \land\ i-1 \in \text{DOMAIN}\ transaction \Rightarrow \\
\qquad\qquad TransactionPhase(transaction[i-1].status) > TransactionPhase(TransactionInitializing) \\
\qquad \land\ \lor\ \land\ transaction[i].targets = \{\} \\
\qquad\qquad \land\ \lor\ \land\ transaction[i].type = TransactionChange \\
\qquad\qquad\qquad \land\ transaction' = [transaction\ \text{EXCEPT}\ ![i].targets = \text{DOMAIN}\ transaction[i].values] \\
\qquad\qquad\qquad \land\ proposal' = [t \in \text{DOMAIN}\ proposal \mapsto \\
\qquad\qquad\qquad\qquad \text{IF}\ t \in \text{DOMAIN}\ transaction[i].values\ \text{THEN} \\
\qquad\qquad\qquad\qquad\quad proposal[t]\ @@\ (i :> [type \mapsto ProposalChange, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad index \mapsto i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad values \mapsto transaction[i].values[t], \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad prevIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad nextIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad rollbackIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad rollbackValues \mapsto \langle\rangle, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad status \mapsto ProposalInitializing]) \\
\qquad\qquad\qquad\qquad \text{ELSE} \\
\qquad\qquad\qquad\qquad\quad proposal[t]] \\
\qquad\qquad\quad \lor\ \land\ transaction[i].type = TransactionRollback \\
\qquad\qquad\qquad \land\ \lor\ \land\ transaction[i].rollback \in \text{DOMAIN}\ transaction \\
\qquad\qquad\qquad\qquad \land\ transaction[transaction[i].rollback].type = TransactionChange \\
\qquad\qquad\qquad\qquad \land\ transaction' = [transaction\ \text{EXCEPT}\ ![i].targets = \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{DOMAIN}\ transaction[transaction[i].rollback].values] \\
\qquad\qquad\qquad\qquad \land\ proposal' = [t \in \text{DOMAIN}\ proposal \mapsto \\
\qquad\qquad\qquad\qquad\qquad \text{IF}\ t \in \text{DOMAIN}\ transaction[transaction[i].rollback].values\ \text{THEN} \\
\qquad\qquad\qquad\qquad\qquad\quad proposal[t]\ @@\ (i :> [type \mapsto ProposalRollback, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad index \mapsto i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad rollback \mapsto transaction[i].rollback, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad prevIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad nextIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad rollbackIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad rollbackValues \mapsto \langle\rangle, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad status \mapsto ProposalInitializing])
\end{array}
$$

```
                                ELSE
                                  proposal[t]]
                        ∨ ∧ ∨ ∧ transaction[i].rollback ∈ DOMAIN transaction
                              ∧ transaction[transaction[i].rollback].type = TransactionRollback
                            ∨ transaction[i].rollback ∉ DOMAIN transaction
                          ∧ transaction' = [transaction EXCEPT ![i].status = TransactionFailed]
                          ∧ UNCHANGED ⟨proposal⟩
              ∨ ∧ transaction[i].targets ≠ {}
                ∧ ∨ ∧ ∀ t ∈ transaction[i].targets : proposal[t][i].status = ProposalInitialized
                    ∧ transaction' = [transaction EXCEPT ![i].status = TransactionInitialized]
                    ∧ UNCHANGED ⟨proposal⟩
                  ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status = ProposalFailed
                    ∧ transaction' = [transaction EXCEPT ![i].status = TransactionFailed]
                    ∧ UNCHANGED ⟨proposal⟩
∨ ∧ transaction[i].status = TransactionInitialized
  ∧ ∀ t ∈ transaction[i].targets :
      proposal[t][i].prevIndex ≠ 0 ⇒
        (transaction[proposal[t][i].prevIndex].isolation = IsolationSerializable ⇒
          TransactionPhase(transaction[proposal[t][i].prevIndex].status) ≥
            TransactionPhase(TransactionValidated))
  ∧ transaction' = [transaction EXCEPT ![i].status = TransactionValidating]
  ∧ UNCHANGED ⟨proposal⟩
∨ ∧ transaction[i].status = TransactionValidating
  ∧ ∨ ∧ ∃ t ∈ transaction[i].targets :
          ProposalPhase(proposal[t][i].status) < ProposalPhase(ProposalValidating)
      ∧ proposal' = [t ∈ DOMAIN proposal ↦
                      IF t ∈ transaction[i].targets THEN
                        [proposal[t] EXCEPT ![i].status = ProposalValidating]
                      ELSE
                        proposal[t]]
      ∧ UNCHANGED ⟨transaction⟩
    ∨ ∧ ∀ t ∈ transaction[i].targets : proposal[t][i].status = ProposalValidated
      ∧ transaction' = [transaction EXCEPT ![i].status = TransactionValidated]
      ∧ UNCHANGED ⟨proposal⟩
    ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status = ProposalFailed
      ∧ transaction' = [transaction EXCEPT ![i].status = TransactionFailed]
      ∧ UNCHANGED ⟨proposal⟩
∨ ∧ transaction[i].status = TransactionValidated
  ∧ ∀ t ∈ transaction[i].targets :
      proposal[t][i].prevIndex ≠ 0 ⇒
        (transaction[proposal[t][i].prevIndex].isolation = IsolationSerializable ⇒
          TransactionPhase(transaction[proposal[t][i].prevIndex].status) ≥
            TransactionPhase(TransactionCommitted))
  ∧ transaction' = [transaction EXCEPT ![i].status = TransactionCommitting]
  ∧ UNCHANGED ⟨proposal⟩
```

```
    ∨ ∧ transaction[i].status = TransactionCommitting
      ∧ ∨ ∧ ∃ t ∈ transaction[i].targets :
                ProposalPhase(proposal[t][i].status) < ProposalPhase(ProposalCommitting)
          ∧ proposal' = [t ∈ DOMAIN proposal ↦
                          IF t ∈ transaction[i].targets THEN
                            [proposal[t] EXCEPT ![i].status = ProposalCommitting]
                          ELSE
                            proposal[t]]
          ∧ UNCHANGED ⟨transaction⟩
        ∨ ∧ ∀ t ∈ transaction[i].targets : proposal[t][i].status = ProposalCommitted
          ∧ transaction' = [transaction EXCEPT ![i].status = TransactionCommitted]
          ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status = ProposalFailed
          ∧ transaction' = [transaction EXCEPT ![i].status = TransactionFailed]
          ∧ UNCHANGED ⟨proposal⟩
    ∨ ∧ transaction[i].status = TransactionCommitted
      ∧ ∀ t ∈ transaction[i].targets :
          proposal[t][i].prevIndex ≠ 0 ⇒
            (transaction[proposal[t][i].prevIndex].isolation = IsolationSerializable ⇒
              TransactionPhase(transaction[proposal[t][i].prevIndex].status) ≥
                TransactionPhase(TransactionApplied))
      ∧ transaction' = [transaction EXCEPT ![i].status = TransactionApplying]
      ∧ UNCHANGED ⟨proposal⟩
    ∨ ∧ transaction[i].status = TransactionApplying
      ∧ ∨ ∧ ∃ t ∈ transaction[i].targets :
                ProposalPhase(proposal[t][i].status) < ProposalPhase(ProposalApplying)
          ∧ proposal' = [t ∈ DOMAIN proposal ↦
                          IF t ∈ transaction[i].targets THEN
                            [proposal[t] EXCEPT ![i].status = ProposalApplying]
                          ELSE
                            proposal[t]]
          ∧ UNCHANGED ⟨transaction⟩
        ∨ ∧ ∀ t ∈ transaction[i].targets : proposal[t][i].status = ProposalApplied
          ∧ transaction' = [transaction EXCEPT ![i].status = TransactionApplied]
          ∧ UNCHANGED ⟨proposal⟩
        ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status = ProposalFailed
          ∧ transaction' = [transaction EXCEPT ![i].status = TransactionFailed]
          ∧ UNCHANGED ⟨proposal⟩
      ∨ ∧ transaction[i].status = TransactionApplied
  ∧ UNCHANGED ⟨configuration, mastership, target⟩

Reconcile a proposal
ReconcileProposal(n, t, i) ≜
  ∧ ∨ ∧ proposal[t][i].status = ProposalInitializing
      ∧ ∨ ∧ configuration[t].proposedIndex > 0
```

$$
\begin{array}{l}
\quad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } \\
\qquad\qquad ![i] = [status \quad\;\; \mapsto ProposalInitialized, \\
\qquad\qquad\qquad\quad prevIndex \mapsto configuration[t].proposedIndex] \mathbin{@@} proposal[t][i], \\
\qquad\qquad ![configuration[t].proposedIndex] = [nextIndex \mapsto i] \mathbin{@@} \\
\qquad\qquad\qquad proposal[t][configuration[t].proposedIndex]]] \\
\quad \lor \land configuration[t].proposedIndex = 0 \\
\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalInitialized]] \\
\land configuration' = [configuration \text{ EXCEPT } ![t].proposedIndex = i] \\
\land \text{UNCHANGED } \langle target \rangle \\
\lor \land proposal[t][i].status = ProposalValidating \\
\land configuration[t].committedIndex = proposal[t][i].prevIndex \\
\land \lor \land proposal[t][i].type = ProposalChange \\
\qquad \land \text{LET } rollbackIndex \triangleq configuration[t].configIndex \\
\qquad\quad\;\; rollbackValues \triangleq [p \in \text{DOMAIN } proposal[t][i].values \mapsto [\\
\qquad\qquad\qquad\qquad p \mapsto \text{IF } p \in \text{DOMAIN } configuration[t].config \text{ THEN} \\
\qquad\qquad\qquad\qquad\qquad configuration[t].values[p] \\
\qquad\qquad\qquad\qquad\quad \text{ELSE} \\
\qquad\qquad\qquad\qquad\qquad [delete \mapsto \text{TRUE}]]] \\
\qquad \text{IN} \quad \exists r \in \{Valid, Invalid\} : \\
\qquad\quad \lor \land r = Valid \\
\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [\\
\qquad\qquad\qquad\qquad proposal[t] \text{ EXCEPT } ![i].rollbackIndex = rollbackIndex, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].rollbackValues = rollbackValues, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].status \quad\;\; = ProposalValidated]] \\
\qquad\quad \lor \land r = Invalid \\
\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [\\
\qquad\qquad\qquad\qquad proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]] \\
\quad \lor \land proposal[t][i].type = ProposalRollback \\
\qquad \land \lor \land configuration[t].index = proposal[t][i].rollback \\
\qquad\qquad \land \lor \land proposal[t][proposal[t][i].rollback].type = ProposalChange \\
\qquad\qquad\qquad \land \text{LET } rollbackIndex \triangleq proposal[t][proposal[t][i].rollback].rollbackIndex \\
\qquad\qquad\qquad\qquad\;\; rollbackValues \triangleq proposal[t][proposal[t][i].rollback].rollbackValues \\
\qquad\qquad\qquad\qquad \text{IN} \quad \exists r \in \{Valid, Invalid\} : \\
\qquad\qquad\qquad\qquad\quad \lor \land r = Valid \\
\qquad\qquad\qquad\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [\\
\qquad\qquad\qquad\qquad\qquad\quad proposal[t] \text{ EXCEPT } ![i].rollbackIndex = rollbackIndex, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![i].rollbackValues = rollbackValues, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![i].status \quad\;\; = ProposalValidated]] \\
\qquad\qquad\qquad\qquad\quad \lor \land r = Invalid \\
\qquad\qquad\qquad\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [\\
\qquad\qquad\qquad\qquad\qquad\qquad proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]] \\
\qquad\qquad \lor \land proposal[t][proposal[t][i].rollabck].type = ProposalRollback \\
\qquad\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].committedIndex = i] \\
\qquad\qquad\quad \land proposal' = [proposal \text{ EXCEPT } ![t] = [\\
\qquad\qquad\qquad proposal[t] \text{ EXCEPT } ![i].status \;\; = ProposalFailed]]
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad\qquad \lor \land configuration[t].index \neq proposal[t][i].rollback \\
\qquad\qquad\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].committedIndex = i] \\
\qquad\qquad\qquad\quad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]] \\
\qquad\quad \land \text{UNCHANGED } \langle target \rangle \\
\qquad \lor \land proposal[t][i].status = ProposalCommitting \\
\qquad\quad \land configuration[t].committedIndex = proposal[t][i].prevIndex \\
\qquad\quad \land \lor \land proposal[t][i].type = ProposalChange \\
\qquad\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].values = proposal[t][i].values, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![t].configIndex = i, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![t].committedIndex = i] \\
\qquad\qquad \lor \land proposal[t][i].type = ProposalRollback \\
\qquad\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].values = proposal[t][i].rollbackValues, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![t].configIndex = proposal[t][i].rollbackIndex, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![t].committedIndex = i] \\
\qquad\quad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalCommitted]] \\
\qquad\quad \land \text{UNCHANGED } \langle target \rangle \\
\qquad \lor \land proposal[t][i].status = ProposalApplying \\
\qquad\quad \land configuration[t].appliedIndex = proposal[t][i].prevIndex \\
\qquad\quad \land configuration[t].appliedTerm = mastership[t].term \\
\qquad\quad \land mastership[t].master = n \\
\qquad\quad \land \exists\, r \in \{Success, Failure\} : \\
\qquad\qquad\quad \lor \land r = Success \\
\qquad\qquad\qquad \land target' = [target \text{ EXCEPT } ![t] = proposal[t][i].values \;@@\; target[t]] \\
\qquad\qquad\qquad \land configuration' = [configuration \text{ EXCEPT } \\
\qquad\qquad\qquad\qquad ![t].appliedIndex = i, \\
\qquad\qquad\qquad\qquad ![t].appliedValues = proposal[t][i].values \;@@\; configuration[i].appliedValues] \\
\qquad\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalApplied]] \\
\qquad\qquad\quad \lor \land r = Failure \\
\qquad\qquad\qquad \land configuration' = [configuration \text{ EXCEPT } ![t].appliedIndex = i] \\
\qquad\qquad\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]] \\
\quad \land \text{UNCHANGED } \langle transaction, mastership \rangle
\end{array}
$$

This section models the Configuration reconciler.

$$
\begin{array}{l}
ReconcileConfiguration(n, t) \triangleq \\
\quad \land \lor \land Target[t].persistent \\
\qquad\quad \land configuration[t].status \neq ConfigurationPersisted \\
\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].status = ConfigurationPersisted] \\
\qquad\quad \land \text{UNCHANGED } \langle target \rangle \\
\qquad \lor \land \neg Target[t].persistent \\
\qquad\quad \land mastership[t].term > configuration[t].configTerm \\
\qquad\quad \land configuration' = [configuration \text{ EXCEPT } ![t].configTerm = mastership[t].term, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![t].status = ConfigurationSynchronizing] \\
\qquad\quad \land \text{UNCHANGED } \langle target \rangle
\end{array}
$$

$$
\begin{array}{l}
\quad \lor \land \neg Target[t].persistent \\
\qquad \land configuration[t].status \neq ConfigurationUnknown \\
\qquad \land mastership[t].term = configuration[t].configTerm \\
\qquad \land mastership[t].master = Nil \\
\qquad \land configuration' = [configuration \text{ EXCEPT } ![t].status = ConfigurationUnknown] \\
\qquad \land \text{UNCHANGED } \langle target \rangle \\
\quad \lor \land configuration[t].status = ConfigurationSynchronizing \\
\qquad \land mastership[t].term = configuration[t].configTerm \\
\qquad \land mastership[t].master = n \\
\qquad \land target' = [target \text{ EXCEPT } ![t] = configuration[t].values] \\
\qquad \land configuration' = [configuration \text{ EXCEPT } ![t].appliedTerm = mastership[t].term, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![t].status \quad\;\; = ConfigurationSynchronized] \\
\land \text{UNCHANGED } \langle proposal,\ transaction,\ mastership \rangle
\end{array}
$$

Init and next state predicates

$$
\begin{array}{l}
Init \triangleq \\
\quad \land transaction = \langle\rangle \\
\quad \land proposal = [t \in \text{DOMAIN } Target \mapsto \\
\qquad\qquad\qquad [p \in \{\} \mapsto [status \quad\;\; \mapsto ProposalInitializing]]] \\
\quad \land configuration = [t \in \text{DOMAIN } Target \mapsto \\
\qquad\qquad\qquad\qquad [target \mapsto t, \\
\qquad\qquad\qquad\qquad\; status \mapsto ConfigurationUnknown, \\
\qquad\qquad\qquad\qquad\; values \mapsto \\
\qquad\qquad\qquad\qquad\quad [path \in \{\} \mapsto \\
\qquad\qquad\qquad\qquad\qquad\; [path \mapsto path, \\
\qquad\qquad\qquad\qquad\qquad\; value \mapsto Nil, \\
\qquad\qquad\qquad\qquad\qquad\; index \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\; deleted \mapsto \text{FALSE}]], \\
\qquad\qquad\qquad\qquad\; configIndex \quad\;\; \mapsto 0, \\
\qquad\qquad\qquad\qquad\; configTerm \quad\;\; \mapsto 0, \\
\qquad\qquad\qquad\qquad\; proposedIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\; committedIndex \mapsto 0, \\
\qquad\qquad\qquad\qquad\; appliedIndex \quad\;\; \mapsto 0, \\
\qquad\qquad\qquad\qquad\; appliedTerm \quad\;\; \mapsto 0, \\
\qquad\qquad\qquad\qquad\; appliedValues \quad \mapsto \\
\qquad\qquad\qquad\qquad\quad [path \in \{\} \mapsto \\
\qquad\qquad\qquad\qquad\qquad\; [path \mapsto path, \\
\qquad\qquad\qquad\qquad\qquad\; value \mapsto Nil, \\
\qquad\qquad\qquad\qquad\qquad\; index \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\; deleted \mapsto \text{FALSE}]]]] \\
\quad \land target = [t \in \text{DOMAIN } Target \mapsto \\
\qquad\qquad\qquad [path \in \{\} \mapsto \\
\qquad\qquad\qquad\quad [value \mapsto Nil]]]
\end{array}
$$

$\land\ mastership = [t \in \text{DOMAIN}\ Target \mapsto [master \mapsto Nil,\ term \mapsto 0]]$

$Next \triangleq$
$\lor \exists\, c \in ValidChanges :$
$Change(c)$
$\lor \exists\, t \in \text{DOMAIN}\ transaction :$
$Rollback(t)$
$\lor \exists\, n \in Node :$
$\exists\, t \in \text{DOMAIN}\ Target :$
$SetMaster(n,\ t)$
$\lor \exists\, t \in \text{DOMAIN}\ Target :$
$UnsetMaster(t)$
$\lor \exists\, n \in Node :$
$\exists\, t \in \text{DOMAIN}\ transaction :$
$ReconcileTransaction(n,\ t)$
$\lor \exists\, n \in Node :$
$\exists\, t \in \text{DOMAIN}\ proposal :$
$\exists\, i \in \text{DOMAIN}\ proposal[t] :$
$ReconcileProposal(n,\ t,\ i)$
$\lor \exists\, n \in Node :$
$\exists\, c \in \text{DOMAIN}\ configuration :$
$ReconcileConfiguration(n,\ c)$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Order \triangleq \text{TRUE}$ *TODO* redefine order spec

THEOREM $Safety \triangleq Spec \Rightarrow \Box Order$

$Completion \triangleq \forall\, i \in \text{DOMAIN}\ transaction :$
$transaction[i].status \in \{TransactionApplied,\ TransactionFailed\}$

THEOREM $Liveness \triangleq Spec \Rightarrow \Diamond Completion$

* Modification History
* Last modified Sun *Feb* 06 02:16:54 *PST* 2022 by *jordanhalterman*
* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*