```
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

TransactionStatus ≜
   ⟨TransactionInitializing,
    TransactionInitialized,
    TransactionValidating,
    TransactionValidated,
    TransactionCommitting,
    TransactionCommitted,
    TransactionApplying,
    TransactionApplied,
    TransactionFailed⟩
 Proposal type constants
CONSTANTS
   ProposalChange,
   ProposalRollback

 Proposal status constants
CONSTANTS
   ProposalInitializing,
   ProposalInitialized,
   ProposalValidating,
   ProposalValidated,
   ProposalCommitting,
   ProposalCommitted,
   ProposalApplying,
   ProposalApplied,
   ProposalFailed

ProposalStatus ≜
    ⟨ProposalInitializing,
    ProposalInitialized,
    ProposalValidating,
    ProposalValidated,
    ProposalCommitting,
    ProposalCommitted,
    ProposalApplying,
    ProposalApplied,
    ProposalFailed⟩

 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: $Target \triangleq [$
$\quad target1 \mapsto [\, persistent \mapsto \text{FALSE},\ values \mapsto [$
$\qquad path1 \mapsto \{\text{“}value1\text{”},\ \text{“}value2\text{”}\},$
$\qquad path2 \mapsto \{\text{“}value2\text{”},\ \text{“}value3\text{”}\}]],$
$\quad target2 \mapsto [\, persistent \mapsto \text{TRUE},\ values \mapsto [$
$\qquad path2 \mapsto \{\text{“}value3\text{”},\ \text{“}value4\text{”}\},$
$\qquad path3 \mapsto \{\text{“}value4\text{”},\ \text{“}value5\text{”}\}]]]$

$\text{CONSTANT}\ Target$

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

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

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

$\text{ASSUME}\ Nil \in \text{STRING}$

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

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

$\text{ASSUME}\ ConfigurationUnknown \in \text{STRING}$
$\text{ASSUME}\ ConfigurationSynchronizing \in \text{STRING}$
$\text{ASSUME}\ ConfigurationSynchronized \in \text{STRING}$
$\text{ASSUME}\ ConfigurationPersisted \in \text{STRING}$
$\text{ASSUME}\ ConfigurationFailed \in \text{STRING}$

$$\begin{aligned}\text{ASSUME}\ & \land IsFiniteSet(Node) \\ & \land \forall\, n \in Node : \\ & \qquad \land n \notin \text{DOMAIN}\ Target \\ & \qquad \land n \in \text{STRING}\end{aligned}$$

```
ASSUME $\land \forall t \in$ DOMAIN $Target$ :
           $\land t \notin Node$
           $\land t \in$ STRING
           $\land Target[t].persistent \in$ BOOLEAN
           $\land \forall p \in$ 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 $\triangleq$ [
  $type$      $::= type \in TransactionType,$
  $index$     $::= index \in Nat,$
  $isolation ::= isolation \in \{IsolationDefault, IsolationSerializable\}$
  $values ::=$ [
    $target \in$ SUBSET (DOMAIN $Target$) $\mapsto$ [ $path \in$ SUBSET (DOMAIN $Target[target].values$) $\mapsto$
      [
        $value ::= value \in$ STRING,
        $delete ::= delete \in$ BOOLEAN ]]],
  $rollback ::= index \in Nat,$
  $targets ::= targets \in$ SUBSET (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 ::= index ∈ 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 ≜ ⟨transaction, proposal, configuration, mastership, target⟩
```

---

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) ≜
    ∧ mastership[t].master ≠ n
    ∧ mastership′ = [mastership EXCEPT ![t].term   = mastership[t].term + 1,
                                       ![t].master = n]
    ∧ UNCHANGED ⟨transaction, proposal, configuration, target⟩

UnsetMaster(t) ≜
    ∧ mastership[t].master ≠ Nil
    ∧ mastership′ = [mastership EXCEPT ![t].master = Nil]
    ∧ UNCHANGED ⟨transaction, proposal, configuration, target⟩
```

---

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

```
Value(s, t, p) ≜
    LET value ≜ CHOOSE v ∈ s : v.target = t ∧ v.path = p
    IN
       [value  ↦ value.value,
        delete ↦ value.delete]

Paths(s, t) ≜
    [p ∈ {v.path : v ∈ {v ∈ s : v.target = t}} ↦ Value(s, t, p)]

Changes(s) ≜
    [t ∈ {v.target : v ∈ s} ↦ Paths(s, t)]

ValidValues(t, p) ≜
    UNION {{[value ↦ v, delete ↦ FALSE] : v ∈ Target[t].values[p]}, {[value ↦ Nil, delete ↦ TRUE]}}

ValidPaths(t) ≜
    UNION {{v @@ [path ↦ p] : v ∈ ValidValues(t, p)} : p ∈ DOMAIN Target[t].values}

ValidTargets ≜
    UNION {{p @@ [target ↦ t] : p ∈ ValidPaths(t)} : t ∈ 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.
$ValidChanges \triangleq$
  $\text{LET } changeSets \triangleq \{s \in \text{SUBSET } ValidTargets :$
                      $\forall t \in \text{DOMAIN } Target :$
                        $\forall p \in \text{DOMAIN } Target[t].values :$
                          $Cardinality(\{v \in s : v.target = t \land v.path = p\}) \leq 1\}$
  $\text{IN}$
    $\{Changes(s) : s \in changeSets\}$

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.
$NextIndex \triangleq$
  $\text{IF DOMAIN } transaction = \{\} \text{ THEN}$
    $1$
  $\text{ELSE}$
    $\text{LET } i \triangleq \text{CHOOSE } i \in \text{DOMAIN } transaction :$
          $\forall j \in \text{DOMAIN } transaction : i \geq j$
    $\text{IN} \quad i + 1$

Add a set of changes 'c' to the transaction log
$Change(c) \triangleq$
  $\land \quad \exists isolation \in \{IsolationDefault, IsolationSerializable\} :$
        $\land transaction' = transaction \mathbin{@@} (NextIndex :> [type \mapsto TransactionChange,$
                                                      $index \mapsto NextIndex,$
                                                      $isolation \mapsto isolation,$
                                                      $values \mapsto c,$
                                                      $targets \mapsto \{\},$
                                                      $status \mapsto TransactionInitializing])$
  $\land \quad \text{UNCHANGED } \langle proposal, configuration, mastership, target \rangle$

Add a rollback of transaction 't' to the transaction log
$Rollback(t) \triangleq$
  $\land \exists isolation \in \{IsolationDefault, IsolationSerializable\} :$
      $\land transaction' = transaction \mathbin{@@} (NextIndex :> [type \mapsto TransactionRollback,$
                                                    $index \mapsto NextIndex,$
                                                    $isolation \mapsto isolation,$
                                                    $rollback \mapsto t,$
                                                    $targets \mapsto \{\},$
                                                    $status \mapsto TransactionInitializing])$
  $\land \text{UNCHANGED } \langle proposal, configuration, mastership, target \rangle$
```

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

```
ReconcileTransaction(n, i) ≜
   ∧ ∨ ∧ transaction[i].status = TransactionInitializing
       ∧ i − 1 ∈ DOMAIN transaction ⇒
             TransactionPhase(transaction[i − 1].status) > TransactionPhase(TransactionInitializing)
       ∧ ∨ ∧ transaction[i].targets = {}
           ∧ ∨ ∧ transaction[i].type = TransactionChange
               ∧ transaction′ = [transaction EXCEPT ![i].targets = DOMAIN transaction[i].values]
               ∧ proposal′ = [t ∈ DOMAIN proposal ↦ proposal[t] @@
                                  (i :> [type   ↦ ProposalChange,
                                         index  ↦ i,
                                         values ↦ transaction[i].changes[t],
                                         status ↦ ProposalInitializing])]
             ∨ ∧ transaction[i].type = TransactionRollback
               ∧ ∨ ∧ transaction[i].rollback ∈ DOMAIN transaction
                   ∧ transaction[transaction[i].rollback].type = TransactionChange
                   ∧ transaction′ = [transaction EXCEPT ![i].targets =
                                        DOMAIN transaction[transaction[i].rollback].values]
                   ∧ proposal′ = [t ∈ DOMAIN proposal ↦ proposal[t] @@
                                      (i :> [type     ↦ ProposalRollback,
                                             index    ↦ i,
                                             rollback ↦ transaction[i].rollback,
                                             status   ↦ ProposalInitializing])]
                 ∨ ∧ ∨ ∧ 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 = ProposalFailed
                   ∧ transaction′ = [transaction EXCEPT ![i].status = TransactionFailed]
             ∨ ∧ ∀ t ∈ transaction[i].targets :
                   ∧ proposal[t][i].status = ProposalInitialized
                   ∧ transaction′ = [transaction EXCEPT ![i].status = TransactionInitialized]
```

```
∨ ∧ 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]
```

```
            $\land \textsc{unchanged}\ \langle proposal \rangle$
      $\lor \land transaction[i].status = TransactionCommitted$
        $\land \forall t \in transaction[i].targets :$
            $proposal[t][i].prevIndex \neq 0 \Rightarrow$
              $(transaction[proposal[t][i].prevIndex].isolation = IsolationSerializable \Rightarrow$
                $TransactionPhase(transaction[proposal[t][i].prevIndex].status) \geq$
                  $TransactionPhase(TransactionApplied))$
        $\land transaction' = [transaction\ \textsc{except}\ ![i].status = TransactionApplying]$
        $\land \textsc{unchanged}\ \langle proposal \rangle$
      $\lor \land transaction[i].status = TransactionApplying$
        $\land \lor \land \exists t \in transaction[i].targets :$
                  $ProposalPhase(proposal[t][i].status) < ProposalPhase(ProposalApplying)$
            $\land proposal' = [t \in \textsc{domain}\ proposal \mapsto$
                                $\textsc{if}\ t \in transaction[i].targets\ \textsc{then}$
                                  $[proposal[t]\ \textsc{except}\ ![i].status = ProposalApplying]$
                                 $\textsc{else}$
                                  $proposal[t]]$
            $\land \textsc{unchanged}\ \langle transaction \rangle$
          $\lor \land \forall t \in transaction[i].targets : proposal[t][i].status = ProposalApplied$
            $\land transaction' = [transaction\ \textsc{except}\ ![i].status = TransactionApplied]$
            $\land \textsc{unchanged}\ \langle proposal \rangle$
          $\lor \land \exists t \in transaction[i].targets : proposal[t][i].status = ProposalFailed$
            $\land transaction' = [transaction\ \textsc{except}\ ![i].status = TransactionFailed]$
            $\land \textsc{unchanged}\ \langle proposal \rangle$
      $\lor \land transaction[i].status = TransactionApplied$
    $\land \textsc{unchanged}\ \langle configuration, mastership, target \rangle$
```

Reconcile a proposal

```
$ReconcileProposal(n, t, i) \triangleq$
    $\land \lor \land proposal[t][i].status = ProposalInitializing$
        $\land \lor \land configuration[t].proposedIndex > 0$
              $\land proposal' = [proposal\ \textsc{except}\ ![t] = [proposal[t]\ \textsc{except}$
                                 $![i] = [status \mapsto ProposalInitialized,$
                                          $prevIndex \mapsto configuration[t].proposedIndex]\ @@\ proposal[t][i],$
                                 $![configuration[t].proposedIndex] = [nextIndex \mapsto i]\ @@$
                                        $proposal[t][configuration[t].proposedIndex]]]$
            $\lor \land configuration[t].proposedIndex = 0$
              $\land proposal' = [proposal\ \textsc{except}\ ![t] = [proposal[t]\ \textsc{except}\ ![i].status = ProposalInitialized]]$
        $\land configuration' = [configuration\ \textsc{except}\ ![t].proposedIndex = i]$
        $\land \textsc{unchanged}\ \langle target \rangle$
      $\lor \land proposal[t][i].status = ProposalValidating$
        $\land configuration[t].committedIndex = proposal[t][i].prevIndex$
        $\land \lor \land proposal[t][i].type = ProposalChange$
              $\land \textsc{let}\ rollbackIndex \triangleq configuration[t].configIndex$
                      $rollbackValues \triangleq [p \in \textsc{domain}\ proposal[t][i].values \mapsto [$
```

```
                                    p ↦ IF p ∈ DOMAIN configuration[t].config THEN
                                            configuration[t].values[p]
                                          ELSE
                                            [delete ↦ TRUE]]]
                IN   ∃ r ∈ {Valid, Invalid} :
                       ∨ ∧ r = Valid
                         ∧ proposal′ = [proposal EXCEPT ![t] = [
                                 proposal[t] EXCEPT ![i].rollbackIndex  = rollbackIndex,
                                                    ![i].rollbackValues = rollbackValues,
                                                    ![i].status         = ProposalValidated]]
                       ∨ ∧ r = Invalid
                         ∧ proposal′ = [proposal EXCEPT ![t] = [
                                 proposal[t] EXCEPT ![i].status = ProposalFailed]]
            ∨ ∧ proposal[t][i].type = ProposalRollback
              ∧ ∨ ∧ configuration[t].index = proposal[t][i].rollback
                  ∧ ∨ ∧ proposal[t][proposal[t][i].rollback].type = ProposalChange
                      ∧ LET rollbackIndex  ≜ proposal[t][proposal[t][i].rollback].rollbackIndex
                            rollbackValues ≜ proposal[t][proposal[t][i].rollback].rollbackValues
                        IN   ∃ r ∈ {Valid, Invalid} :
                               ∨ ∧ r = Valid
                                 ∧ proposal′ = [proposal EXCEPT ![t] = [
                                      proposal[t] EXCEPT ![i].rollbackIndex  = rollbackIndex,
                                                         ![i].rollbackValues = rollbackValues,
                                                         ![i].status         = ProposalValidated]]
                               ∨ ∧ r = Invalid
                                 ∧ proposal′ = [proposal EXCEPT ![t] = [
                                          proposal[t] EXCEPT ![i].status = ProposalFailed]]
                    ∨ ∧ proposal[t][proposal[t][i].rollabck].type = ProposalRollback
                      ∧ configuration′ = [configuration EXCEPT ![t].committedIndex = i]
                      ∧ proposal′ = [proposal EXCEPT ![t] = [
                          proposal[t] EXCEPT ![i].status  = ProposalFailed]]
                ∨ ∧ configuration[t].index ≠ proposal[t][i].rollback
                  ∧ configuration′ = [configuration EXCEPT ![t].committedIndex = i]
                  ∧ proposal′ = [proposal EXCEPT ![t] = [proposal[t] EXCEPT ![i].status = ProposalFailed]]
          ∧ UNCHANGED ⟨target⟩
        ∨ ∧ proposal[t][i].status = ProposalCommitting
          ∧ configuration[t].committedIndex = proposal[t][i].prevIndex
          ∧ ∨ ∧ proposal[t][i].type = ProposalChange
              ∧ configuration′ = [configuration EXCEPT ![t].values         = proposal[t][i].values,
                                                       ![t].configIndex    = i,
                                                       ![t].committedIndex = i]
            ∨ ∧ proposal[t][i].type = ProposalRollback
              ∧ configuration′ = [configuration EXCEPT ![t].values         = proposal[t][i].rollbackValues,
                                                       ![t].configIndex    = proposal[t][i].rollbackIndex,
                                                       ![t].committedIndex = i]
          ∧ proposal′ = [proposal EXCEPT ![t] = [proposal[t] EXCEPT ![i].status = ProposalCommitted]]
          ∧ UNCHANGED ⟨target⟩
       ∨ ∧ proposal[t][i].status = ProposalApplying
          ∧ configuration[t].appliedIndex = proposal[t][i].prevIndex
          ∧ configuration[t].appliedTerm = mastership[t].term
          ∧ mastership[t].master = n
          ∧ ∃ r ∈ {Success, Failure} :
               ∨ ∧ r = Success
                  ∧ target′ = [target EXCEPT ![t] = proposal[t][i].values @@ target[t]]
                  ∧ configuration′ = [configuration EXCEPT
                        ![t].appliedIndex = i,
                        ![t].appliedValues = proposal[t][i].values @@ configuration[i].appliedValues]
                  ∧ proposal′ = [proposal EXCEPT ![t] = [proposal[t] EXCEPT ![i].status = ProposalApplied]]
               ∨ ∧ r = Failure
                  ∧ configuration′ = [configuration EXCEPT ![t].appliedIndex = i]
                  ∧ proposal′ = [proposal EXCEPT ![t] = [proposal[t] EXCEPT ![i].status = ProposalFailed]]
    ∧ UNCHANGED ⟨transaction, mastership⟩
```

---

This section models the Configuration reconciler.

```
ReconcileConfiguration(n, t) ≜
    ∧ ∨ ∧ target[t].persistent
          ∧ configuration[t].status ≠ ConfigurationPersisted
          ∧ configuration′ = [configuration EXCEPT ![t].status = ConfigurationPersisted]
          ∧ UNCHANGED ⟨target⟩
       ∨ ∧ ¬target[t].persistent
          ∧ mastership[t].term > configuration[t].term
          ∧ configuration′ = [configuration EXCEPT ![t].term    = mastership[t].term,
                                                   ![t].status  = ConfigurationSynchronizing]
          ∧ UNCHANGED ⟨target⟩
       ∨ ∧ ¬target[t].persistent
          ∧ configuration[t].status ≠ ConfigurationUnknown
          ∧ mastership[t].term = configuration[t].term
          ∧ mastership[t].master = Nil
          ∧ configuration′ = [configuration EXCEPT ![t].status = ConfigurationUnknown]
          ∧ UNCHANGED ⟨target⟩
       ∨ ∧ configuration[t].status = ConfigurationSynchronizing
          ∧ mastership[t].master = n
          ∧ target′ = [target EXCEPT ![t] = configuration[t].values]
          ∧ configuration′ = [configuration EXCEPT ![t].appliedTerm = mastership[t].term,
                                                   ![t].status      = ConfigurationSynchronized]
    ∧ UNCHANGED ⟨proposal, transaction, mastership⟩
```

---

*Init* and next state predicates

```
Init ≜
    ∧ transaction = ⟨⟩
    ∧ proposal = [t ∈ DOMAIN Target ↦
                       [p ∈ {} ↦ [status        ↦ ProposalInitializing]]]
    ∧ configuration = [t ∈ DOMAIN Target ↦
                            [target ↦ t,
                             status ↦ ConfigurationUnknown,
                             values ↦
                                [path ∈ {}  ↦
                                    [path    ↦ path,
                                     value   ↦ Nil,
                                     index   ↦ 0,
                                     deleted ↦ FALSE]],
                             configIndex    ↦ 0,
                             proposedIndex  ↦ 0,
                             committedIndex ↦ 0,
                             appliedIndex   ↦ 0,
                             appliedTerm    ↦ 0,
                             appliedValues  ↦
                                [path ∈ {}  ↦
                                    [path    ↦ path,
                                     value   ↦ Nil,
                                     index   ↦ 0,
                                     deleted ↦ FALSE]]]]
    ∧ target = [t ∈ DOMAIN Target ↦
                     [path ∈ {} ↦
                         [value ↦ Nil]]]
    ∧ mastership = [t ∈ DOMAIN Target ↦ [master ↦ Nil, term ↦ 0]]

Next ≜
    ∨ ∃ c ∈ ValidChanges :
        Change(c)
    ∨ ∃ t ∈ DOMAIN transaction :
        Rollback(t)
    ∨ ∃ n ∈ Node :
        ∃ t ∈ DOMAIN Target :
          SetMaster(n, t)
    ∨ ∃ t ∈ DOMAIN Target :
        UnsetMaster(t)
    ∨ ∃ n ∈ Node :
        ∃ t ∈ DOMAIN transaction :
            ReconcileTransaction(n, t)
    ∨ ∃ n ∈ Node :
        ∃ c ∈ DOMAIN configuration :
```

$\qquad\qquad ReconcileConfiguration(n,\ c)$

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

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

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

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

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

\* Modification History
\* Last modified Sun *Feb* 06 01:55:54 *PST* 2022 by *jordanhalterman*
\* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*