```
──────────────────── 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
    $ProposalInitializing$,
    $ProposalInitialized$,
    $ProposalValidating$,
    $ProposalValidated$,
    $ProposalCommitting$,
    $ProposalCommitted$,
    $ProposalApplying$,
    $ProposalApplied$,
    $ProposalFailed$

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

Configuration status constants

CONSTANTS
    $ConfigurationUnknown$,
    $ConfigurationSynchronizing$,
    $ConfigurationSynchronized$,
    $ConfigurationPersisted$,
    $ConfigurationFailed$

The set of all nodes

CONSTANT $Node$

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

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

```
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 $\land \forall t \in$ DOMAIN $Target :$
           $\land t \notin Node$
           $\land t \in$ STRING
           $\land Target[t].persistent \in$ BOOLEAN
           $\land \forall p \in$ DOMAIN $Target[t].paths :$
               $IsFiniteSet(Target[t].paths[p])$
```

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 ∈
  {TransactionChange,
   TransactionRollback}

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

TYPE Transaction ≜ [
  type      ::= type ∈ TransactionType,
  index     ::= index ∈ Nat,
  isolation ::= isolation ∈ {IsolationDefault, IsolationSerializable}
  values ::= [
    target ∈ SUBSET (DOMAIN Target) ↦ [ path ∈ SUBSET (DOMAIN Target[target].paths) ↦
      [
        value ::= value ∈ STRING,
        delete ::= delete ∈ BOOLEAN ]]],
  rollback ::= index ∈ Nat,
  targets ::= targets ∈ SUBSET (DOMAIN Target)
  status ::= status ∈ TransactionStatus]

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

```
$ValidChanges \triangleq$
    $\text{LET}\ changeSets \triangleq \{s \in \text{SUBSET}\ ValidTargets :$
                              $\forall t \in \text{DOMAIN}\ Target \quad :$
                                $\forall p \in \text{DOMAIN}\ Target[t] :$
                                  $Cardinality(\{v \in s : v.target = t \land v.path = p\}) \leq 1\}$
```

```
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.

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

Add a set of changes 'c' to the transaction log

```
$Change(c) \triangleq$
  $\land$  $\exists isolation \in \{IsolationDefault, IsolationSerializable\}$ :
        $\land transaction' = transaction @@ (NextIndex :> [type \mapsto TransactionChange,$
                                                        $index \mapsto NextIndex,$
                                                        $isolation \mapsto isolation,$
                                                        $values \mapsto c,$
                                                        $targets \mapsto \{\},$
                                                        $status \mapsto TransactionInitializing])$
  $\land$  UNCHANGED $\langle proposal, configuration, mastership, target\rangle$
```

Add a rollback of transaction 't' to the transaction log

```
$Rollback(t) \triangleq$
  $\land \exists isolation \in \{IsolationDefault, IsolationSerializable\}$ :
      $\land transaction' = transaction @@ (NextIndex :> [type \mapsto TransactionRollback,$
                                                      $index \mapsto NextIndex,$
                                                      $isolation \mapsto isolation,$
                                                      $rollback \mapsto t,$
                                                      $targets \mapsto \{\},$
                                                      $status \mapsto TransactionInitializing])$
  $\land$ 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

```
$ReconcileTransaction(n, i) \triangleq$
    $\land \lor \land transaction[i].status = TransactionInitializing$
            $\land i - 1 \in transaction \Rightarrow$
                    $TransactionPhase(transaction[i-1].status) > TransactionPhase(TransactionInitializing)$
            $\land \lor \land transaction[i].targets = \{\}$
                    $\land \lor \land transaction[i].type = TransactionChange$
                            $\land transaction' = [transaction \text{ EXCEPT } ![i].targets = \text{DOMAIN } transaction[i].values]$
                            $\land proposal' = [t \in \text{DOMAIN } proposal \mapsto proposal[t] @@$
                                                 $(i :> [type \mapsto ProposalChange,$
                                                         $index \mapsto i,$
                                                         $values \mapsto transaction[i].changes[t],$
                                                         $status \mapsto ProposalInitializing])]$
                        $\lor \land transaction[i].type = TransactionRollback$
                            $\land \lor \land transaction[i].rollback \in \text{DOMAIN } transaction$
                                    $\land transaction[transaction[i].rollback].type = TransactionChange$
                                    $\land transaction' = [transaction \text{ EXCEPT } ![i].targets =$
                                                           $\text{DOMAIN } transaction[transaction[i].rollback].values]$
                                    $\land proposal' = [t \in \text{DOMAIN } proposal \mapsto proposal[t] @@$
                                                         $(i :> [type \mapsto ProposalRollback,$
                                                                 $index \mapsto i,$
                                                                 $rollback \mapsto transaction[i].rollback,$
                                                                 $status \mapsto ProposalInitializing])]$
                                $\lor \land \lor \land transaction[i].rollback \in \text{DOMAIN } transaction$
                                            $\land transaction[transaction[i].rollback].type = TransactionRollback$
                                        $\lor transaction[i].rollback \notin \text{DOMAIN } transaction$
                                    $\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionFailed]$
                                    $\land \text{UNCHANGED } \langle proposal \rangle$
                $\lor \land transaction[i].targets \neq \{\}$
                    $\land \lor \land \exists t \in transaction[i].targets :$
                                    $\land proposal[t][i].status = ProposalFailed$
                                    $\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionFailed]$
                        $\lor \land \forall t \in transaction[i].targets :$
                                    $\land proposal[t][i].status = ProposalInitialized$
                                    $\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionInitialized]$
        $\lor \land transaction[i].status = TransactionInitialized$
            $\land \forall t \in transaction[i].targets :$
                $proposal[t][i].prevIndex \neq 0 \Rightarrow$
                    $(transaction[proposal[t][i].prevIndex].isolation = IsolationSerializable \Rightarrow$
                        $TransactionPhase(transaction[proposal[t][i].prevIndex].status) \geq$
                            $TransactionPhase(TransactionValidated))$
            $\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionValidating]$
            $\land \text{UNCHANGED } \langle proposal \rangle$
        $\lor \land transaction[i].status = TransactionValidating$
        ∧ ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status ≠ ProposalValidating
            ∧ proposal′ = [t ∈ DOMAIN proposal ↦
                                 IF t ∈ transaction[i].targets THEN
                                    [proposal[t] EXCEPT ![i].status = ProposalValidating]
                                  ELSE
                                    proposal[t]]
            ∧ UNCHANGED ⟨transaction⟩
      ∨ ∧ 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 : proposal[t][i].status ≠ ProposalCommitting
            ∧ proposal′ = [t ∈ DOMAIN proposal ↦
                                 IF t ∈ transaction[i].targets THEN
                                    [proposal[t] EXCEPT ![i].status = ProposalCommitting]
                                  ELSE
                                    proposal[t]]
            ∧ UNCHANGED ⟨transaction⟩
      ∨ ∧ 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]
        ∧ ∨ ∧ ∃ t ∈ transaction[i].targets : proposal[t][i].status ≠ ProposalApplying
            ∧ proposal′ = [t ∈ DOMAIN proposal ↦
                                 IF t ∈ transaction[i].targets THEN
                                    [proposal[t] EXCEPT ![i].status = ProposalApplying]
                                  ELSE
                                    proposal[t]]
            ∧ UNCHANGED ⟨transaction⟩
        ∧ UNCHANGED ⟨proposal⟩
      ∨ ∧ transaction[i].status = TransactionApplying
      ∨ ∧ transaction[i].status = TransactionApplied
   ∧ UNCHANGED ⟨configuration, mastership, target⟩
```

Reconcile a proposal

```
ReconcileProposal(n, t, i) ≜
   ∧ ∨ ∧ proposal[t][i].status = ProposalInitializing
```

```
  $\land \lor \land configuration[t].proposedIndex > 0$
          $\land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT }$
                        $![i] = [status \mapsto ProposalInitialized,$
                                 $prevIndex \mapsto configuration[t].proposedIndex] \;@@\; proposal[t][i],$
                        $![configuration[t].proposedIndex] = [nextIndex \mapsto i] \;@@$
                              $proposal[t][configuration[t].proposedIndex]]]$
      $\lor \land configuration[t].proposedIndex = 0$
          $\land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalInitialized]]$
  $\land configuration' = [configuration \text{ EXCEPT } ![t].proposedIndex = i]$
  $\land \text{UNCHANGED } \langle target \rangle$
$\lor \land proposal[t][i].status = ProposalValidating$
  $\land configuration[t].committedIndex = proposal[t][i].prevIndex$
  $\land \lor \land proposal[t][i].type = ProposalChange$
          $\land \text{LET } rollbackIndex \triangleq configuration[t].configIndex$
                 $rollbackValues \triangleq [p \in \text{DOMAIN } proposal[t][i].values \mapsto [$
                                 $p \mapsto \text{IF } p \in \text{DOMAIN } configuration[t].config \text{ THEN}$
                                        $configuration[t].values[p]$
                                      $\text{ELSE}$
                                        $[delete \mapsto \text{TRUE}]]]$
            $\text{IN}$
              $\land proposal' = [proposal \text{ EXCEPT } ![t] = [$
                                  $proposal[t] \text{ EXCEPT } ![i].rollbackIndex = rollbackIndex,$
                                                       $![i].rollbackValues = rollbackValues,$
                                                       $![i].status = ProposalValidated]]$
      $\lor \land proposal[t][i].type = ProposalRollback$
          $\land \lor \land configuration[t].index = proposal[t][i].rollback$
                  $\land \lor \land proposal[t][i].rollback \in \text{DOMAIN } proposal[t]$
                          $\land \lor \land proposal[t][proposal[t][i].rollback].type = ProposalChange$
                                  $\land \text{LET } rollbackIndex \triangleq proposal[t][proposal[t][i].rollback].rollbackIndex$
                                         $rollbackValues \triangleq proposal[t][proposal[t][i].rollback].rollbackValues$
                                    $\text{IN} \quad proposal' = [proposal \text{ EXCEPT } ![t] = [$
                                                   $proposal[t] \text{ EXCEPT } ![i].rollbackIndex = rollbackIndex,$
                                                                        $![i].rollbackValues = rollbackValues,$
                                                                        $![i].status = ProposalValidated]$
                              $\lor \land proposal[t][proposal[t][i].rollabck].type = ProposalRollback$
                                  $\land configuration' = [configuration \text{ EXCEPT } ![t].committedIndex = i]$
                                  $\land proposal' = [proposal \text{ EXCEPT } ![t] = [$
                                        $proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]]$
                      $\lor \land proposal[t][i].rollback \notin \text{DOMAIN } proposal[t]$
                          $\land configuration' = [configuration \text{ EXCEPT } ![t].committedIndex = i]$
                          $\land proposal' = [proposal \text{ EXCEPT } ![t] = [$
                                $proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]]$
              $\lor \land configuration[t].index \neq proposal[t][i].rollback$
                  $\land configuration' = [configuration \text{ EXCEPT } ![t].committedIndex = i]$
                  $\land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].status = ProposalFailed]]$
          ∧ 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
          ∧ target′ = [target EXCEPT ![t] = proposal[t][i].values @@ target[t]]
          ∧ proposal′ = [proposal EXCEPT ![t] = [proposal[t] EXCEPT ![i].status = ProposalApplied]]
    ∧ 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]
    $\land$ UNCHANGED $\langle proposal, transaction, mastership \rangle$
```

Init and next state predicates

```
$Init \triangleq$
    $\land\ transaction = \langle\rangle$
    $\land\ proposal = [t \in$ DOMAIN $Target \mapsto$
                        $[p \in \{\} \mapsto [status \mapsto ProposalInitializing]]]$
    $\land\ configuration = [t \in$ DOMAIN $Target \mapsto$
                            $[target \mapsto t,$
                             $status \mapsto ConfigurationUnknown,$
                             $values \mapsto$
                                $[path \in \{\} \mapsto$
                                    $[path \mapsto path,$
                                     $value \mapsto Nil,$
                                     $index \mapsto 0,$
                                     $deleted \mapsto$ FALSE$]],$
                             $configIndex \mapsto 0,$
                             $proposedIndex \mapsto 0,$
                             $committedIndex \mapsto 0,$
                             $appliedIndex \mapsto 0,$
                             $appliedTerm \mapsto 0,$
                             $appliedValues \mapsto$
                                $[path \in \{\} \mapsto$
                                    $[path \mapsto path,$
                                     $value \mapsto Nil,$
                                     $index \mapsto 0,$
                                     $deleted \mapsto$ FALSE$]]]]$
    $\land\ target = [t \in$ DOMAIN $Target \mapsto$
                    $[path \in \{\} \mapsto$
                        $[value \mapsto Nil]]]$
    $\land\ mastership = [t \in$ DOMAIN $Target \mapsto [master \mapsto Nil, term \mapsto 0]]$

$Next \triangleq$
    $\lor\ \exists\, c \in ValidChanges :$
        $Change(c)$
    $\lor\ \exists\, t \in$ DOMAIN $transaction :$
        $Rollback(t)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, t \in$ DOMAIN $Target :$
          $SetMaster(n, t)$
    $\lor\ \exists\, t \in$ DOMAIN $Target :$
        $UnsetMaster(t)$
    $\lor\ \exists\, n \in Node :$
        $\exists\, t \in$ DOMAIN $transaction :$
```

```
            $ReconcileTransaction(n, t)$
    $\lor \exists n \in Node :$
        $\exists c \in \text{DOMAIN } configuration :$
            $ReconcileConfiguration(n, c)$
```

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

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

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

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

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

\* Modification History
\* Last modified Sun *Feb* 06 01:23:24 *PST* 2022 by *jordanhalterman*
\* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*