```
─────────────────── MODULE Config ───────────────────
INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

LOCAL INSTANCE TLC

─────────────────────────────────────────────────────

 An empty constant
CONSTANT Nil

 Transaction type constants
CONSTANTS
    Change,
    Rollback

 Transaction isolation constants
CONSTANTS
    ReadCommitted,
    Serializable

 Phase constants
CONSTANTS
    Initialize,
    Validate,
    Abort,
    Commit,
    Apply

Phase ≜
    {Initialize,
     Validate,
     Abort,
     Commit,
     Apply}

 Status constants
CONSTANTS
    InProgress,
    Complete,
    Failed

State ≜
    {InProgress,
     Complete,
    Failed}
```

State constants

```
CONSTANTS
    Pending,
    Validated,
    Committed,
    Applied,
    Aborted

Status ≜
    {Pending,
     Validated,
     Committed,
     Applied,
     Aborted}

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 ≜
    [target1 ↦
       [persistent ↦ FALSE, values ↦ [
          path1 ↦ {"value1", "value2"},
          path2 ↦ {"value2", "value3"}]],
     target2 ↦
       [persistent ↦ TRUE, values ↦ [
          path2 ↦ {"value3", "value4"},
          path3 ↦ {"value4", "value5"}]]]
```

```
CONSTANT Target
```

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 Type ::= *type* ∈

```
  {Change,
   Rollback}

TYPE Phase ::= phase ∈
  {Initialize,
   Validate,
   Abort,
   Commit,
   Apply}

TYPE State ::= state ∈
  {InProgress,
   Complete,
   Failed}

TYPE Status ::= status ∈
  {Pending,
   Validated,
   Committed,
   Applied,
   Aborted}

TYPE Isolation ::= isolation ∈
  {ReadCommitted,
   Serializable}

TYPE Transaction ≜
  [type      ::= type ∈  Type,
   isolation ::= isolation ∈ Isolation
   change ::=
     [target ∈ SUBSET  (DOMAIN  Target)  ↦
        [path  ∈ SUBSET  (DOMAIN  Target[target].values)  ↦
           [value  ::=  value  ∈ STRING,
            delete  ::=  delete  ∈ BOOLEAN ]]],
   rollback ::=  index  ∈ Nat,
   targets ::= targets ∈ SUBSET  (DOMAIN  Target)
   phase     ::= phase ∈ Phase,
   state     ::= state ∈ State,
   status ::= status ∈ Status]

TYPE Proposal ≜
  [type      ::= type ∈  Type,
   change     ::=
     [index  ::=  index  ∈ Nat,
      values ::=
        [path  ∈ SUBSET  (DOMAIN  Target[target].values)  ↦
           [value  ::=  value  ∈ STRING,
            delete  ::=  delete  ∈ BOOLEAN ]]],
   rollback ::=
     [index  ::=  index  ∈ Nat,
      values ::=
        [path  ∈ SUBSET  (DOMAIN  Target[target].values)  ↦
           [value  ::=  value  ∈ STRING,
            delete  ::=  delete  ∈ BOOLEAN ]]],
    dependency  ::=  [index  ∈ Nat],
    phase       ::= phase ∈ Phase,
    state       ::= state ∈ State]

 TYPE Configuration ≜
   [config ::=
     [index  ::=  index  ∈ Nat,
      term  ::=  term  ∈ Nat,
      values ::=
        [path  ∈ SUBSET  (DOMAIN Target[target])  ↦
           [value  ::=  value  ∈ STRING,
            index  ::=  index  ∈ Nat,
            deleted  ::=  delete  ∈ BOOLEAN ]]],
    proposal ::=  [index  ::=  index  ∈ Nat],
    commit  ::=  [index  ::=  index  ∈ Nat],
    target ::=
      [index  ::=  index  ∈ Nat,
       term  ::=  term  ∈ Nat,
       values ::=
         [path  ∈ SUBSET  (DOMAIN Target[target])  ↦
            [value  ::=  value  ∈ STRING,
             index  ::=  index  ∈ Nat,
             deleted  ::=  delete  ∈ BOOLEAN ]]],
    state ::= state ∈ State]

 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⟩

├──────────────────────────────────────────────────────────────┤

Transaction   ≜ INSTANCE Transaction
Proposal      ≜ INSTANCE Proposal
Configuration ≜ INSTANCE Configuration
Southbound     ≜ INSTANCE Southbound
Northbound    ≜ INSTANCE Northbound
```

Formal specification, constraints, and theorems.

```
Init ≜
    ∧ Transaction!Init
    ∧ Proposal!Init
    ∧ Configuration!Init
    ∧ Northbound!Init
    ∧ Southbound!Init

Next ≜
    ∨ ∧ Transaction!Next
      ∧ UNCHANGED ⟨configuration, target, mastership⟩
    ∨ ∧ Proposal!Next
      ∧ UNCHANGED ⟨transaction⟩
    ∨ ∧ Configuration!Next
      ∧ UNCHANGED ⟨transaction, proposal⟩
    ∨ ∧ Northbound!Next
      ∧ UNCHANGED ⟨proposal, configuration, target, mastership⟩
    ∨ ∧ Southbound!Next
      ∧ UNCHANGED ⟨transaction, proposal, configuration⟩

Spec ≜ Init ∧ □[Next]_vars ∧ WF_vars(Next)

Order ≜
   ∀ t ∈ DOMAIN proposal :
     ∀ i ∈ DOMAIN proposal[t] :
        ∧ ∧ proposal[t][i].phase = Commit
          ∧ proposal[t][i].state  = InProgress
          ⇒ ¬∃ j ∈ DOMAIN proposal[t] :
                  ∧ j > i
                  ∧ proposal[t][j].phase = Commit
                  ∧ proposal[t][j].state  = Complete
        ∧ ∧ proposal[t][i].phase = Apply
          ∧ proposal[t][i].state  = InProgress
          ⇒ ¬∃ j ∈ DOMAIN proposal[t] :
                  ∧ j > i
                  ∧ proposal[t][j].phase = Apply
                  ∧ proposal[t][j].state  = Complete

Consistency ≜
   ∀ t ∈ DOMAIN target :
     LET
          Compute the transaction indexes that have been applied to the target
         targetIndexes ≜ {i ∈ DOMAIN transaction :
                                  ∧ i ∈ DOMAIN proposal[t]
```

```
                        $\land\ proposal[t][i].phase = Apply$
                        $\land\ proposal[t][i].state\ = Complete$
                        $\land\ t \in transaction[i].targets$
                        $\land\ \neg\exists\, j \in \text{DOMAIN}\ transaction :$
                              $\land\ j > i$
                              $\land\ transaction[j].type = Rollback$
                              $\land\ transaction[j].rollback = i$
                              $\land\ transaction[j].phase = Apply$
                              $\land\ transaction[j].state\ = Complete\}$
         Compute the set of paths in the target that have been updated by transactions
        $appliedPaths\ \ \triangleq\ \text{UNION}\ \{\text{DOMAIN}\ proposal[t][i].change.values : i \in targetIndexes\}$
         Compute the highest index applied to the target for each path
        $pathIndexes\ \ \ \triangleq\ [p \in appliedPaths \mapsto \text{CHOOSE}\ i \in targetIndexes :$
                               $\forall\, j \in targetIndexes :$
                                  $\land\ i\ \geq j$
                                  $\land\ p \in \text{DOMAIN}\ proposal[t][i].change.values]$
         Compute the expected target configuration based on the last indexes applied
         to the target for each path.
        $expectedConfig\ \triangleq\ [p \in \text{DOMAIN}\ pathIndexes \mapsto proposal[t][pathIndexes[p]].change.values[p]]$
      IN
        $target[t] = expectedConfig$

$Isolation\ \triangleq$
  $\forall\, i \in \text{DOMAIN}\ transaction :$
    $\land\ \land\ transaction[i].phase = Commit$
       $\land\ transaction[i].state\ = InProgress$
       $\land\ transaction[i].isolation = Serializable$
       $\Rightarrow \neg\exists\, j \in \text{DOMAIN}\ transaction :$
             $\land\ j > i$
             $\land\ transaction[j].targets \cap transaction[i].targets \neq \{\}$
             $\land\ transaction[j].phase = Commit$
    $\land\ \land\ transaction[i].phase = Apply$
       $\land\ transaction[i].state\ = InProgress$
       $\land\ transaction[i].isolation = Serializable$
       $\Rightarrow \neg\exists\, j \in \text{DOMAIN}\ transaction :$
             $\land\ j > i$
             $\land\ transaction[j].targets \cap transaction[i].targets \neq \{\}$
             $\land\ transaction[j].phase = Apply$

$Safety\ \triangleq\ \Box(Order \land Consistency \land Isolation)$

THEOREM $Spec \Rightarrow Safety$

$Terminated(i)\ \triangleq$
   $\land\ i \in \text{DOMAIN}\ transaction$
   $\land\ transaction[i].phase \in \{Apply,\ Abort\}$
```

$\quad \land\ \mathit{transaction}[i].\mathit{state}\ = \mathit{Complete}$

$\mathit{Termination}\ \triangleq$
$\quad \forall\, i \in 1\,..\,\mathit{Len}(\mathit{transaction}) : \mathit{Terminated}(i)$

$\mathit{Liveness}\ \triangleq\ \Diamond \mathit{Termination}$

$\textsc{theorem}\ \mathit{Spec} \Rightarrow \mathit{Liveness}$

Type assumptions.

$\textsc{assume}\ \mathit{Nil} \in \textsc{string}$

$\textsc{assume}\ \forall\, \mathit{phase} \in \mathit{Phase} : \mathit{phase} \in \textsc{string}$

$\textsc{assume}\ \forall\, \mathit{state}\ \in \mathit{State}\ : \mathit{state}\ \in \textsc{string}$

$\textsc{assume}\ \forall\, \mathit{status} \in \mathit{Status} : \mathit{status} \in \textsc{string}$

```
ASSUME  $\land\ \mathit{IsFiniteSet}(\mathit{Node})$
        $\land\ \forall\, n \in \mathit{Node} :$
            $\land\ n \notin \textsc{domain}\ \mathit{Target}$
            $\land\ n \in \textsc{string}$
```

```
ASSUME  $\land\ \forall\, t \in \textsc{domain}\ \mathit{Target} :$
            $\land\ t \notin \mathit{Node}$
            $\land\ t \in \textsc{string}$
            $\land\ \mathit{Target}[t].\mathit{persistent} \in \textsc{boolean}$
            $\land\ \forall\, p \in \textsc{domain}\ \mathit{Target}[t].\mathit{values} :$
                $\mathit{IsFiniteSet}(\mathit{Target}[t].\mathit{values}[p])$
```

\* Modification History
\* Last modified Sun *Feb* 20 08:03:14 *PST* 2022 by *jordanhalterman*
\* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*