```
MODULE Transaction

INSTANCE Naturals

INSTANCE FiniteSets

INSTANCE Sequences

INSTANCE TLC

  An empty constant
CONSTANT Nil

  Transaction phase constants
CONSTANTS
    Change,
    Rollback

  Proposal phase constants
CONSTANTS
    Commit,
    Apply

  Status constants
CONSTANTS
    Pending,
    Complete,
    Aborted,
    Failed

Status ≜ {Pending, Complete, Aborted, Failed}

  The set of all nodes
CONSTANT Node

  The set of possible paths and values
CONSTANT Path, Value

Empty ≜ [p ∈ {} ↦ Nil]

  Variables defined by other modules.
VARIABLES
    configuration,
    mastership,
    conn,
    target
```

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

```
VARIABLE transaction
```

A sequence of configuration changes used for model checking.

```
VARIABLE history

TypeOK ≜
   ∀ i ∈ DOMAIN transaction :
      ∧ transaction[i].type ∈ {Change, Rollback}
      ∧ transaction[i].index ∈ Nat
      ∧ transaction[i].revision ∈ Nat
      ∧ transaction[i].change.index ∈ Nat
      ∧ transaction[i].change.revision ∈ Nat
      ∧ ∀ p ∈ DOMAIN transaction[i].change.values :
            transaction[i].change.values[p] ≠ Nil ⇒
               transaction[i].change.values[p] ∈ STRING
      ∧ transaction[i].rollback.index ∈ Nat
      ∧ transaction[i].rollback.revision ∈ Nat
      ∧ ∀ p ∈ DOMAIN transaction[i].rollback.values :
            transaction[i].rollback.values[p] ≠ Nil ⇒
               transaction[i].rollback.values[p] ∈ STRING
      ∧ transaction[i].commit ∈ Status
      ∧ transaction[i].apply ∈ Status

LOCAL State ≜ [
   transactions  ↦ [i ∈ DOMAIN transaction ↦ transaction[i] @@ [index ↦ i]],
   configuration ↦ configuration]

LOCAL Transitions ≜
   LET
      transactions ≜ {i ∈ DOMAIN transaction′ :
                           i ∈ DOMAIN transaction ⇒ transaction′[i] ≠ transaction[i]}
   IN
     [transactions ↦ [i ∈ transactions ↦ transaction′[i] @@ [index ↦ i]]]

Test ≜ INSTANCE Test WITH
   File ← "Transaction.log"
```

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

```
LOCAL Transaction(i) ≜
   IF i ∈ DOMAIN transaction THEN
      transaction[i]
    ELSE [
    $index \mapsto i,$
    $revision \mapsto 0,$
    $change \mapsto [$
        $index \mapsto 0,$
        $revision \mapsto 0],$
    $rollback \mapsto [$
        $index \mapsto 0,$
        $revision \mapsto 0],$
    $commit \mapsto Nil,$
    $apply \mapsto Nil]$

LOCAL $LastTransaction \triangleq Transaction(Len(transaction))$
```

CHANGE [$index = 1$, $revision = 1$, $change = (index = 1, revision = 1)$, $rollback = (index = 0, revision = 0)$] ← − *Change* revision 1
CHANGE [$index = 2$, $revision = 2$, $change = (index = 2, revision = 2)$, $rollback = (index = 1, revision = 1)$]
CHANGE [$index = 3$, $revision = 3$, $change = (index = 3, revision = 3)$, $rollback = (index = 2, revision = 2)$]
ROLLBACK [$index = 4$, $revision = 3$, $change = (index = 2, revision = 2)$, $rollback = (index = 3, revision = 3)$] ← − Roll back revision 3 at index 3, leading to revision 2
ROLLBACK [$index = 5$, $revision = 3$, $change = (index = 1, revision = 1)$, $rollback = (index = 2, revision = 2)$]
CHANGE [$index = 6$, $revision = 4$, $change = (index = 6, revision = 4)$, $rollback = (index = 1, revision = 1)$]
CHANGE [$index = 7$, $revision = 5$, $change = (index = 7, revision = 5)$, $rollback = (index = 6, revision = 4)$]
ROLLBACK [$index = 8$, $revision = 5$, $change = (index = 6, revision = 4)$, $rollback = (index = 7, revision = 5)$] ← − Roll back revision 5 at index 7, leading to revision 4
ROLLBACK [$index = 9$, $revision = 5$, $change = (index = 1, revision = 1)$, $rollback = (index = 6, revision = 4)$] ← − Roll back revision 4 at index 6, leading to revision 1 CHANGE [$index = 10$, $revision = 6$, $change = (index = 10, revision = 6)$, $rollback = (index = 1, revision = 1)$]

Add a change for revision 'i' to the transaction log

```
$AppendChange(i) \triangleq$
    $\land LastTransaction.revision = i - 1$
    $\land Len(transaction) > 0 \Rightarrow transaction[Len(transaction)].commit = Complete$
    $\land \exists p \in Path, v \in Value :$
        $\land transaction' = Append(transaction, [$
                $type \mapsto Change,$
                $index \mapsto Len(transaction) + 1,$
                $revision \mapsto i,$
                $change \mapsto [$
                    $index \mapsto Len(transaction) + 1,$
                    $revision \mapsto i,$
                    $values \mapsto (p :> v)],$
                $rollback \mapsto [$
                    $index \mapsto LastTransaction.change.index,$
```

```
                                   $revision \mapsto LastTransaction.change.revision,$
                                   $values \;\;\; \mapsto \text{IF } p \in \text{DOMAIN } configuration.committed.values \text{ THEN}$
                                               $(p :> configuration.committed.values[p])$
                                             $\text{ELSE}$
                                               $(p :> Nil)],$
                                   $commit \;\; \mapsto Pending,$
                                   $apply \;\;\; \mapsto Pending])$
    $\land \text{UNCHANGED } \langle configuration, mastership, conn, target, history \rangle$
```

Add a rollback of revision 'i' to the transaction log

```
$RollbackChange(i) \triangleq$
    $\land LastTransaction.change.revision = i$
    $\land Len(transaction) > 0 \Rightarrow transaction[Len(transaction)].commit = Complete$
    $\land transaction' = Append(transaction, [$
                          $type \;\;\;\; \mapsto Rollback,$
                          $index \;\;\; \mapsto Len(transaction) + 1,$
                          $revision \mapsto LastTransaction.revision,$
                          $change \;\; \mapsto [$
                             $index \;\;\;\; \mapsto transaction[LastTransaction.change.index].rollback.index,$
                             $revision \;\; \mapsto transaction[LastTransaction.change.index].rollback.revision,$
                             $values \;\;\; \mapsto transaction[LastTransaction.change.index].rollback.values],$
                          $rollback \mapsto [$
                             $index \;\;\;\; \mapsto LastTransaction.change.index,$
                             $revision \;\; \mapsto i,$
                             $values \;\;\; \mapsto Empty],$
                          $commit \;\;\; \mapsto Pending,$
                          $apply \;\;\;\; \mapsto Pending])$
    $\land \text{UNCHANGED } \langle configuration, mastership, conn, target, history \rangle$
```

```
$CommitChange(n, i) \triangleq$
    $\land transaction[i].commit = Pending$
    $\land i - 1 \in \text{DOMAIN } transaction \Rightarrow$
            $transaction[i - 1].commit \neq Pending$
    $\land configuration' = [configuration \text{ EXCEPT } !.committed.index \;\;\;\; = transaction[i].change.index,$
                                                   $!.committed.revision = transaction[i].change.revision,$
                                                   $!.committed.values \;\;\; = transaction[i].change.values \; @@$
                                                                              $configuration.committed.values]$
    $\land transaction' = [transaction \text{ EXCEPT } ![i].commit = Complete]$
    $\land history' = Append(history, [$
                       $type \;\;\;\; \mapsto Change,$
                       $phase \;\;\; \mapsto Commit,$
                       $revision \mapsto transaction[i].change.revision])$
    $\land \text{UNCHANGED } \langle target \rangle$
```

```
$ApplyChange(n, i) \triangleq$
    $\land\ transaction[i].apply = Pending$
    $\land\ transaction[i].commit = Complete$
    $\land\ i - 1 \in \textsc{domain}\ transaction \Rightarrow$
            $transaction[i-1].apply \neq Pending$
    $\land\ \lor\ \land\ i - 1 \in \textsc{domain}\ transaction \Rightarrow$
                    $transaction[i-1].apply = Complete$
          $\land\ configuration.state = Complete$
          $\land\ configuration.term = mastership.term$
          $\land\ conn[n].id = mastership.conn$
          $\land\ conn[n].connected$
          $\land\ target.running$
              Apply to the target successfully.
          $\land\ \lor\ \land\ target' = [target\ \textsc{except}\ !.values = transaction[i].change.values\ @@\ target.values]$
                $\land\ configuration' = [configuration\ \textsc{except}\ !.applied.index = transaction[i].change.index,$
                                                                 $!.applied.revision = transaction[i].change.revision,$
                                                                 $!.applied.values = transaction[i].change.values\ @@$
                                                                                        $configuration.applied.values]$
                $\land\ transaction' = [transaction\ \textsc{except}\ ![i].apply = Complete]$
                $\land\ history' = Append(history, [$
                                  $type \mapsto Change,$
                                  $phase \mapsto Apply,$
                                  $revision \mapsto transaction[i].change.revision])$
              Apply to the target failed.
             $\lor\ \land\ transaction' = [transaction\ \textsc{except}\ ![i].apply = Failed]$
                $\land\ \textsc{unchanged}\ \langle configuration, target, history \rangle$
       $\lor\ \land\ i - 1 \in \textsc{domain}\ transaction$
          $\land\ transaction[i-1].apply \in \{Aborted, Failed\}$
          $\land\ transaction' = [transaction\ \textsc{except}\ ![i].apply = Aborted]$
          $\land\ \textsc{unchanged}\ \langle configuration, target, history \rangle$

$ReconcileChange(n, i) \triangleq$
    $\land\ transaction[i].type = Change$
    $\land\ \lor\ CommitChange(n, i)$
       $\lor\ ApplyChange(n, i)$

$CommitRollback(n, i) \triangleq$
    $\land\ transaction[i].commit = Pending$
    $\land\ i - 1 \in \textsc{domain}\ transaction \Rightarrow$
            $transaction[i-1].commit \neq Pending$
    $\land\ configuration' = [configuration\ \textsc{except}\ !.committed.index = transaction[i].change.index,$
                                                   $!.committed.revision = transaction[i].change.revision,$
                                                   $!.committed.values = transaction[i].change.values\ @@$
                                                                            $configuration.committed.values]$
    $\land\ transaction' = [transaction\ \textsc{except}\ ![i].commit = Complete]$
    $\land\ history' = Append(history, [$
                          $type \quad\ \mapsto Rollback,$
                          $phase \quad \mapsto Commit,$
                          $revision \mapsto transaction[i].rollback.revision])$
    $\land\ \textsc{unchanged}\ \langle target \rangle$

$ApplyRollback(n, i)\ \triangleq$
    $\land\ transaction[i].apply = Pending$
    $\land\ transaction[i].commit = Complete$
    $\land\ i - 1 \in \textsc{domain}\ transaction \Rightarrow$
            $transaction[i - 1].apply \neq Pending$
    $\land\ \lor\ \land\ transaction[transaction[i].rollback.index].apply \in \{Complete, Failed\}$
          $\land\ configuration.state = Complete$
          $\land\ configuration.term = mastership.term$
          $\land\ conn[n].id = mastership.conn$
          $\land\ conn[n].connected$
          $\land\ target.running$
          $\land\ target' = [target\ \textsc{except}\ !.values = transaction[i].change.values\ @@\ target.values]$
          $\land\ configuration' = [configuration\ \textsc{except}\ !.applied.index \quad\ = transaction[i].change.index,$
                                                  $!.applied.revision\ = transaction[i].change.revision,$
                                                  $!.applied.values \quad = transaction[i].change.values\ @@$
                                                                        $configuration.applied.values]$
          $\land\ transaction' = [transaction\ \textsc{except}\ ![i].apply = Complete]$
          $\land\ history' = Append(history, [$
                                $type \quad\ \mapsto Rollback,$
                                $phase \quad \mapsto Apply,$
                                $revision \mapsto transaction[i].rollback.revision])$
       $\lor\ \land\ transaction[transaction[i].rollback.index].apply = Aborted$
          $\land\ transaction' = [transaction\ \textsc{except}\ ![i].apply = Aborted]$
          $\land\ \textsc{unchanged}\ \langle configuration, target, history \rangle$

$ReconcileRollback(n, i)\ \triangleq$
    $\land\ transaction[i].type = Rollback$
    $\land\ \lor\ CommitRollback(n, i)$
       $\lor\ ApplyRollback(n, i)$

$ReconcileTransaction(n, i)\ \triangleq$
    $\land\ i \in \textsc{domain}\ transaction$
    $\land\ \lor\ ReconcileChange(n, i)$
       $\lor\ ReconcileRollback(n, i)$
    $\land\ \textsc{unchanged}\ \langle mastership, conn \rangle$
```