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

$Status \triangleq \{Pending, Complete, Aborted, Failed\}$

The set of all nodes
CONSTANT *Node*

The set of possible paths and values
CONSTANT *Path*, *Value*

$Empty \triangleq [p \in \{\} \mapsto Nil]$

Variables defined by other modules.
VARIABLES
proposals,
configuration

A transaction log. Transactions may either request a set

of changes to a set of targets or rollback a prior change.

VARIABLE *transactions*

$$
\begin{array}{l}
\mathit{TypeOK} \triangleq \\
\quad \forall\, i \in \text{DOMAIN } \mathit{transactions} : \\
\qquad \land\ \mathit{transactions}[i].\mathit{index} \in \mathit{Nat} \\
\qquad \land\ \mathit{transactions}[i].\mathit{phase} \in \{\mathit{Change},\ \mathit{Rollback}\} \\
\qquad \land\ \mathit{transactions}[i].\mathit{change} \in \mathit{Nat} \\
\qquad \land\ \mathit{transactions}[i].\mathit{rollback} \in \mathit{Nat} \\
\qquad \land\ \forall\, p \in \text{DOMAIN } \mathit{transactions}[i].\mathit{values} : \\
\qquad\quad \mathit{transactions}[i].\mathit{values}[p] \neq \mathit{Nil} \Rightarrow \mathit{transactions}[i].\mathit{values}[p] \in \text{STRING}
\end{array}
$$

$$
\begin{array}{l}
\text{LOCAL } \mathit{State} \triangleq [\\
\quad \mathit{transactions} \mapsto \mathit{transactions}, \\
\quad \mathit{proposals} \mapsto \mathit{proposals}, \\
\quad \mathit{configuration} \mapsto \mathit{configuration}]
\end{array}
$$

$$
\begin{array}{l}
\text{LOCAL } \mathit{Transitions} \triangleq \\
\quad \text{LET} \\
\qquad \mathit{txIndexes} \triangleq \{i \in \text{DOMAIN } \mathit{transactions}' : \\
\qquad\qquad\qquad i \in \text{DOMAIN } \mathit{transactions} \Rightarrow \mathit{transactions}'[i] \neq \mathit{transactions}[i]\} \\
\qquad \mathit{propIndexes} \triangleq \{i \in \text{DOMAIN } \mathit{proposals}' : \\
\qquad\qquad\qquad i \in \text{DOMAIN } \mathit{proposals} \Rightarrow \mathit{proposals}'[i] \neq \mathit{proposals}[i]\} \\
\quad \text{IN} \quad [\mathit{transactions} \mapsto [i \in \mathit{txIndexes} \mapsto \mathit{transactions}'[i]], \\
\qquad\ \mathit{proposals} \mapsto [i \in \mathit{propIndexes} \mapsto \mathit{proposals}'[i]]]
\end{array}
$$

$$
\begin{array}{l}
\mathit{Test} \triangleq \text{INSTANCE } \mathit{Test} \text{ WITH} \\
\quad \mathit{File} \leftarrow \text{“Transaction.log”}
\end{array}
$$

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

$$
\begin{array}{l}
\mathit{AppendChange}(i) \triangleq \\
\quad \land\ \mathit{Len}(\mathit{transactions}) = i - 1 \\
\quad \land\ \exists\, p \in \mathit{Path},\ v \in \mathit{Value} : \\
\qquad \land\ \mathit{transactions}' = \mathit{Append}(\mathit{transactions}, [\\
\qquad\qquad\qquad\qquad \mathit{index} \mapsto \mathit{Len}(\mathit{transactions}) + 1, \\
\qquad\qquad\qquad\qquad \mathit{phase} \mapsto \mathit{Change}, \\
\qquad\qquad\qquad\qquad \mathit{values} \mapsto (p :> v), \\
\qquad\qquad\qquad\qquad \mathit{change} \mapsto 0, \\
\qquad\qquad\qquad\qquad \mathit{rollback} \mapsto 0]) \\
\quad \land\ \text{UNCHANGED } \langle \mathit{proposals},\ \mathit{configuration} \rangle
\end{array}
$$

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

$$
\begin{array}{l}
\mathit{RollbackChange}(i) \triangleq \\
\quad \land\ i \in \text{DOMAIN } \mathit{transactions} \\
\quad \land\ \mathit{transactions}[i].\mathit{phase} = \mathit{Change}
\end{array}
$$

```
    ∧ transactions' = [transactions EXCEPT ![i].phase = Rollback]
    ∧ UNCHANGED ⟨proposals, configuration⟩

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

ReconcileChange(n, i) ≜
    ∧ transactions[i].phase = Change
    ∧ transactions[i].change = 0
    ∧ ∨ ∧ Len(proposals) = 0
        ∧ i = 1
      ∨ ∧ Len(proposals) > 0
        ∧ proposals[Len(proposals)].revision = i − 1
        ∧ proposals[Len(proposals)].commit = Complete
    ∧ LET proposal ≜ [
          type ↦ Change,
          index ↦ Len(proposals) + 1,
          revision ↦ i,
          change ↦ [
              index ↦ Len(proposals) + 1,
              revision ↦ i,
              values ↦ transactions[i].values],
          rollback ↦ [
              index ↦ IF Len(proposals) > 0 THEN proposals[Len(proposals)].change.index ELSE 0,
              revision ↦ IF Len(proposals) > 0 THEN proposals[Len(proposals)].change.revision ELSE 0,
              values ↦ [
                  p ∈ DOMAIN transactions[i].values ↦
                      IF p ∈ DOMAIN configuration.committed.values THEN
                          configuration.committed.values[p]
                      ELSE Nil]],
          commit ↦ Pending,
          apply ↦ Pending]
      IN
          ∧ proposals' = Append(proposals, proposal)
          ∧ transactions' = [transactions EXCEPT ![i].change = Len(proposals')]
    ∧ UNCHANGED ⟨configuration⟩

ReconcileRollback(n, i) ≜
    ∧ transactions[i].phase = Rollback
    ∧ Len(proposals) > 0
    ∧ LET lastProposal ≜ proposals[Len(proposals)]
      IN ∧ lastProposal.change.revision = i
         ∧ lastProposal.commit = Complete
         ∧ LET proposal ≜ [
               type ↦ Rollback,
               index ↦ Len(proposals) + 1,
               revision ↦ lastProposal.revision,
```

```
                change  ↦ [
                   index    ↦ proposals[lastProposal.change.index].rollback.index,
                   revision ↦ proposals[lastProposal.change.index].rollback.revision,
                   values   ↦ proposals[lastProposal.change.index].rollback.values],
                rollback ↦ [
                   index    ↦ lastProposal.change.index,
                   revision ↦ i,
                   values   ↦ Empty],
                commit   ↦ Pending,
                  apply  ↦ Pending]
          IN    ∧ proposals' = Append(proposals, proposal)
                ∧ transactions' = [transactions EXCEPT ![i].rollback = Len(proposals')]
    ∧ UNCHANGED ⟨configuration⟩

ReconcileTransaction(n, i) ≜
    ∧ i ∈ DOMAIN transactions
    ∧ ∨ ReconcileChange(n, i)
      ∨ ReconcileRollback(n, i)
```