MODULE *Transaction*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

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

$Phase \triangleq$
{*Initialize*,
Validate,
Abort,
Commit,
Apply}

Status constants
CONSTANTS
InProgress,
Complete,
Failed

$State \triangleq$
{*InProgress*,
Complete,

Failed}

State constants
CONSTANTS
Pending,
Validated,
Committed,
Applied,
Aborted

Status $\triangleq$
{*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 $\triangleq$
[*target*1 $\mapsto$
[persistent $\mapsto$ FALSE, *values* $\mapsto$ [
*path*1 $\mapsto$ {"*value*1", "*value*2"},
*path*2 $\mapsto$ {"*value*2", "*value*3"}]],
*target*2 $\mapsto$
[persistent $\mapsto$ TRUE, *values* $\mapsto$ [
*path*2 $\mapsto$ {"*value*3", "*value*4"},
*path*3 $\mapsto$ {"*value*4", "*value*5"}]]]
CONSTANT *Target*

Empty $\triangleq$ $[p \in \{\} \mapsto [value \mapsto Nil,\ delete \mapsto \text{FALSE}]]$

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*

$$
\begin{array}{l}
Test \triangleq \text{INSTANCE } Test \text{ WITH} \\
\quad File \leftarrow \text{“Transaction.log”}, \\
\quad CurrState \leftarrow [\\
\qquad transactions \mapsto transaction, \\
\qquad proposals \mapsto proposal, \\
\qquad configuration \mapsto configuration, \\
\qquad mastership \mapsto mastership, \\
\qquad target \mapsto target], \\
\quad SuccState \leftarrow [\\
\qquad transactions \mapsto transaction', \\
\qquad proposals \mapsto proposal', \\
\qquad configuration \mapsto configuration', \\
\qquad mastership \mapsto mastership', \\
\qquad target \mapsto target']
\end{array}
$$

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

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

$$
\begin{array}{l}
RequestChange(i,\ c) \triangleq \\
\quad \land\ i = Len(transaction) + 1 \\
\quad \land\ \exists\, isolation \in \{ReadCommitted,\ Serializable\} : \\
\qquad \land\ transaction' = transaction \mathbin{@@} (i :> [type \mapsto Change, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad isolation \mapsto isolation, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad change \mapsto c, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad targets \mapsto \{\}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad phase \mapsto Initialize, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad state \mapsto InProgress, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad status \mapsto Pending]) \\
\quad \land\ \text{UNCHANGED } \langle proposal,\ configuration,\ mastership,\ target \rangle
\end{array}
$$

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

$$
\begin{array}{l}
RequestRollback(i,\ j) \triangleq \\
\quad \land\ i = Len(transaction) + 1
\end{array}
$$

$\land \exists\, isolation \in \{ReadCommitted,\ Serializable\} :$
$\quad \land\ transaction' = transaction\ @@\ (i :> [type \mapsto Rollback,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad isolation \mapsto isolation,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad rollback \mapsto j,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad targets \mapsto \{\},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad phase \mapsto Initialize,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad state \mapsto InProgress,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad status \mapsto Pending])$
$\land$ 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

Transacations proceed through a series of phases:
* *Initialize* - create and link Proposals
* *Validate* - validate changes and rollbacks
* *Commit* - commit changes to Configurations
* *Apply* - commit changes to Targets

Reconcile a transaction
$ReconcileTransaction(n,\ i) \triangleq$

Initialize is the only transaction phase that's globally serialized. While in the Initializing phase, the reconciler checks whether the prior transaction has been Initialized before creating Proposals in the *Initialize* phase. Once all of the transaction's proposals have been Initialized, the transaction will be marked Initialized. If any proposal is *Failed*, the transaction will be marked *Failed* as well.

$\land\ \lor\ \land\ transaction[i].phase = Initialize$
$\quad \land\ \lor\ \land\ transaction[i].state = InProgress$

All prior transaction must be initialized before proceeding to initialize this transaction.

$\quad\quad \land \neg\exists\, j \in$ DOMAIN $transaction :$
$\quad\quad\quad \land\ j < i$
$\quad\quad\quad \land\ transaction[j].phase = Initialize$
$\quad\quad\quad \land\ transaction[j].state\ \ = InProgress$

If the transaction's targets are not yet set, create proposals and add targets to the transaction state.

$\quad\quad \land\ \lor\ \land\ transaction[i].targets = \{\}$

If the transaction is a change, the targets are taken from the change values.

$\quad\quad\quad \land\ \lor\ \land\ transaction[i].type = Change$
$\quad\quad\quad\quad \land\ transaction' = [transaction$ EXCEPT $![i].targets =$ DOMAIN $transaction[i].change]$
$\quad\quad\quad\quad \land\ proposal' = [t \in$ DOMAIN $proposal \mapsto$

$$
\begin{array}{l}
\text{IF } t \in \text{DOMAIN } transaction[i].change \text{ THEN} \\
\quad proposal[t] \mathbin{@@} (i :> [type \mapsto Change, \\
\qquad\qquad\qquad\qquad\quad change \mapsto \\
\qquad\qquad\qquad\qquad\qquad [index \mapsto i, \\
\qquad\qquad\qquad\qquad\qquad\ values \mapsto transaction[i].change[t]], \\
\qquad\qquad\qquad\qquad\quad rollback \mapsto \\
\qquad\qquad\qquad\qquad\qquad [index \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\ values \mapsto Empty], \\
\qquad\qquad\qquad\qquad\quad dependency \mapsto [index \mapsto 0], \\
\qquad\qquad\qquad\qquad\quad phase \mapsto Initialize, \\
\qquad\qquad\qquad\qquad\quad state \mapsto InProgress]) \\
\quad \text{ELSE} \\
\qquad proposal[t]]
\end{array}
$$

If the transaction is a rollback, the targets affected are the targets of the change transaction being rolled back.

$$
\begin{array}{l}
\lor \land transaction[i].type = Rollback \\
\end{array}
$$

If the rollback index is a valid *Change* transaction, initialize proposals for all of the *Change* targets.

$$
\begin{array}{l}
\land \lor \land transaction[i].rollback \in \text{DOMAIN } transaction \\
\qquad \land transaction[transaction[i].rollback].type = Change \\
\qquad \land transaction' = [transaction \text{ EXCEPT } ![i].targets = \\
\qquad\qquad\qquad\qquad \text{DOMAIN } transaction[transaction[i].rollback].change] \\
\qquad \land proposal' = [t \in \text{DOMAIN } proposal \mapsto \\
\qquad\qquad \text{IF } t \in \text{DOMAIN } transaction[transaction[i].rollback].change \text{ THEN} \\
\qquad\qquad\quad proposal[t] \mathbin{@@} (i :> [type \mapsto Rollback, \\
\qquad\qquad\qquad\qquad\qquad\qquad change \mapsto \\
\qquad\qquad\qquad\qquad\qquad\qquad\ [index \mapsto 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\ values \mapsto Empty], \\
\qquad\qquad\qquad\qquad\qquad\qquad rollback \mapsto \\
\qquad\qquad\qquad\qquad\qquad\qquad\ [index \mapsto transaction[i].rollback, \\
\qquad\qquad\qquad\qquad\qquad\qquad\ values \mapsto Empty], \\
\qquad\qquad\qquad\qquad\qquad\qquad dependency \mapsto [index \mapsto 0], \\
\qquad\qquad\qquad\qquad\qquad\qquad phase \mapsto Initialize, \\
\qquad\qquad\qquad\qquad\qquad\qquad state \mapsto InProgress]) \\
\qquad\qquad \text{ELSE} \\
\qquad\qquad\quad proposal[t]]
\end{array}
$$

If the rollback index is not a valid *Change* transaction fail the *Rollback* transaction.

$$
\begin{array}{l}
\lor \land \lor \land transaction[i].rollback \in \text{DOMAIN } transaction \\
\qquad\quad \land transaction[transaction[i].rollback].type = Rollback \\
\qquad \lor transaction[i].rollback \notin \text{DOMAIN } transaction \\
\quad \land transaction' = [transaction \text{ EXCEPT } ![i].state = Failed] \\
\quad \land \text{UNCHANGED } \langle proposal \rangle
\end{array}
$$

If the transaction's proposals have been initialized, check proposals for completion or failures.

$\lor \land transaction[i].targets \neq \{\}$

If all proposals have been *Complete*, mark the transaction *Complete*.

$\land \lor \land \forall\, t \in transaction[i].targets :$
$\land\ proposal[t][i].phase = Initialize$
$\land\ proposal[t][i].state\ = Complete$
$\land\ transaction' = [transaction$ EXCEPT $![i].state = Complete]$
$\land$ UNCHANGED $\langle proposal \rangle$

If any proposal has been *Failed*, mark the transaction *Failed*.

$\lor \land \exists\, t \in transaction[i].targets :$
$\land\ proposal[t][i].phase\ = Initialize$
$\land\ proposal[t][i].state = Failed$
$\land\ transaction' = [transaction$ EXCEPT $![i].state = Failed]$
$\land$ UNCHANGED $\langle proposal \rangle$

Once the transaction has been Initialized, proceed to the *Validate* phase.
If any of the transaction's proposals depend on a *Serializable* transaction,
verify the dependency has been *Validated* to preserve serializability before
moving the transaction to the *Validate* phase.

$\lor \land transaction[i].state = Complete$
$\land \forall\, t \in transaction[i].targets :$
$\land\ proposal[t][i].dependency.index \in$ DOMAIN $transaction$
$\land\ transaction[proposal[t][i].dependency.index].isolation = Serializable$
$\Rightarrow transaction[proposal[t][i].dependency.index].status \in \{Validated, Committed, Applied, A$
$\land\ transaction' = [transaction$ EXCEPT $![i].phase = Validate,$
$![i].state\ = InProgress]$
$\land$ UNCHANGED $\langle proposal \rangle$

If the transaction failed initialization, proceed to the *Abort* phase
to ensure indexes are still updated for the target configurations.

$\lor \land transaction[i].state = Failed$
$\land\ transaction' = [transaction$ EXCEPT $![i].phase = Abort,$
$![i].state\ = InProgress]$
$\land$ UNCHANGED $\langle proposal \rangle$

$\lor \land transaction[i].phase = Validate$
$\land \lor \land transaction[i].state = InProgress$

Move the transaction's proposals to the Validating state

$\land \lor \land \exists\, t \in transaction[i].targets :$
$\land\ proposal[t][i].phase \neq Validate$
$\land\ proposal' = [proposal$ EXCEPT $![t] =$
$[proposal[t]$ EXCEPT $![i].phase = Validate,$
$![i].state\ = InProgress]]$
$\land$ UNCHANGED $\langle transaction \rangle$

If all proposals have been *Complete*, mark the transaction *Complete*.

$\lor \land \forall\, t \in transaction[i].targets :$
$\land\ proposal[t][i].phase\ = Validate$
$\land\ proposal[t][i].state = Complete$
$\land\ transaction' = [transaction$ EXCEPT $![i].state\ = Complete,$

$$![i].status = Validated]$$

$$\land \text{UNCHANGED } \langle proposal \rangle$$

If any proposal has been *Failed*, mark the transaction *Failed*.

$$\lor \land \exists t \in transaction[i].targets : \land proposal[t][i].phase = Validate \land proposal[t][i].state = Failed$$

$$\land transaction' = [transaction \text{ EXCEPT } ![i].state = Failed]$$

$$\land \text{UNCHANGED } \langle proposal \rangle$$

Once the transaction has been *Validated*, proceed to the *Commit* phase.
If any of the transaction's proposals depend on a *Serializable* transaction,
verify the dependency has been *Committed* to preserve serializability before
moving the transaction to the *Commit* phase.

$$\lor \land transaction[i].state = Complete$$

$$\land \forall t \in transaction[i].targets : \land proposal[t][i].dependency.index \in \text{DOMAIN } transaction \land transaction[proposal[t][i].dependency.index].isolation = Serializable \Rightarrow transaction[proposal[t][i].dependency.index].status \in \{Committed, Applied, Aborted\}$$

$$\land transaction' = [transaction \text{ EXCEPT } ![i].phase = Commit, ![i].state = InProgress]$$

$$\land \text{UNCHANGED } \langle proposal \rangle$$

If the transaction failed validation, proceed to the *Abort* phase
to ensure indexes are still updated for the target configurations.

$$\lor \land transaction[i].state = Failed$$

$$\land transaction' = [transaction \text{ EXCEPT } ![i].phase = Abort, ![i].state = InProgress]$$

$$\land \text{UNCHANGED } \langle proposal \rangle$$

$$\lor \land transaction[i].phase = Commit$$

$$\land \lor \land transaction[i].state = InProgress$$

Move the transaction's proposals to the Committing state

$$\land \lor \land \exists t \in transaction[i].targets : \land proposal[t][i].phase \neq Commit \land proposal' = [proposal \text{ EXCEPT } ![t] = [proposal[t] \text{ EXCEPT } ![i].phase = Commit, ![i].state = InProgress]]$$

$$\land \text{UNCHANGED } \langle transaction \rangle$$

If all proposals have been *Complete*, mark the transaction *Complete*.

$$\lor \land \forall t \in transaction[i].targets : \land proposal[t][i].phase = Commit \land proposal[t][i].state = Complete$$

$$\land transaction' = [transaction \text{ EXCEPT } ![i].state = Complete, ![i].status = Committed]$$

$$\land \text{UNCHANGED } \langle proposal \rangle$$

Once the transaction has been *Committed*, proceed to the *Apply* phase.
If any of the transaction's proposals depend on a *Serializable* transaction,
verify the dependency has been *Applied* to preserve serializability before

moving the transaction to the *Apply* phase.

$\lor \land transaction[i].state = Complete$
$\quad \land \forall\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].dependency.index \in \text{DOMAIN } transaction$
$\qquad \land transaction[proposal[t][i].dependency.index].isolation = Serializable$
$\qquad \Rightarrow transaction[proposal[t][i].dependency.index].status \in \{Applied,\ Aborted\}$
$\quad \land transaction' = [transaction \text{ EXCEPT } ![i].phase = Apply,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].state\ = InProgress]$
$\quad \land \text{UNCHANGED } \langle proposal \rangle$
$\lor \land transaction[i].phase = Apply$
$\quad \land transaction[i].state\ = InProgress$

Move the transaction's proposals to the Applying state

$\land \lor \land \exists\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].phase \neq Apply$
$\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] =$
$\qquad\qquad\qquad [proposal[t] \text{ EXCEPT } ![i].phase = Apply,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].state\ = InProgress]]$
$\quad \land \text{UNCHANGED } \langle transaction \rangle$

If all proposals have been *Complete*, mark the transaction *Complete*.

$\lor \land \forall\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].phase\ = Apply$
$\qquad \land proposal[t][i].state = Complete$
$\quad \land transaction' = [transaction \text{ EXCEPT } ![i].state\ = Complete,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].status = Applied]$
$\quad \land \text{UNCHANGED } \langle proposal \rangle$

If any proposal has been *Failed*, mark the transaction *Failed*.

$\lor \land \exists\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].phase\ = Apply$
$\qquad \land proposal[t][i].state = Failed$
$\quad \land transaction' = [transaction \text{ EXCEPT } ![i].state = Failed]$
$\quad \land \text{UNCHANGED } \langle proposal \rangle$

The Aborting state is used to clean up transactions that have failed during the Initializing or Validating phases.

$\lor \land transaction[i].phase = Abort$
$\quad \land transaction[i].state\ = InProgress$

Move the transaction's proposals to the Aborting state

$\land \lor \land \exists\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].phase \neq Abort$
$\qquad \land proposal' = [proposal \text{ EXCEPT } ![t] =$
$\qquad\qquad\qquad [proposal[t] \text{ EXCEPT } ![i].phase = Abort,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![i].state\ = InProgress]]$
$\quad \land \text{UNCHANGED } \langle transaction \rangle$

If all proposals have been *Complete*, mark the transaction *Complete*.

$\lor \land \forall\, t \in transaction[i].targets :$
$\qquad \land proposal[t][i].phase\ = Abort$

$$
\begin{array}{l}
\quad \land\ proposal[t][i].state = Complete \\
\land\ transaction' = [transaction \text{ EXCEPT } ![i].state = Complete, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![i].status = Aborted] \\
\quad \land \text{ UNCHANGED } \langle proposal \rangle \\
\land \text{ UNCHANGED } \langle configuration,\ mastership,\ target \rangle
\end{array}
$$