
MODULE *Transaction*

EXTENDS *Proposal*

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

Transaction type constants

CONSTANTS

TransactionChange,
TransactionRollback

Transaction isolation constants

CONSTANTS

ReadCommitted,
Serializable

Phase constants

CONSTANTS

TransactionInitialize,
TransactionValidate,
TransactionAbort,
TransactionCommit,
TransactionApply

Status constants

CONSTANTS

TransactionInProgress,
TransactionComplete,
TransactionFailed

State constants

CONSTANTS

TransactionPending,
TransactionValidated,
TransactionCommitted,
TransactionApplied,
TransactionAborted

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

VARIABLE *transaction*

$\text{LOCAL } \textit{InitState} \triangleq$
 $\quad [\textit{transactions} \mapsto \textit{transaction},$
 $\quad \quad \textit{proposals} \mapsto [t \in \text{DOMAIN } \textit{proposal} \mapsto \textit{proposal}[t]]]$

$\text{LOCAL } \textit{NextState} \triangleq$
 $\quad [\textit{transactions} \mapsto \textit{transaction'},$
 $\quad \quad \textit{proposals} \mapsto \textit{proposal'}]$

$\text{LOCAL } \textit{Trace} \triangleq \text{INSTANCE } \textit{Trace} \text{ WITH}$
 $\quad \textit{Module} \leftarrow \text{"Transaction"},$
 $\quad \textit{InitState} \leftarrow \textit{InitState},$
 $\quad \textit{NextState} \leftarrow \textit{NextState}$

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

Transactions 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

$\text{ReconcileTransaction}(i) \triangleq$

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.

$\wedge \vee \wedge \textit{transaction}[i].\textit{phase} = \textit{TransactionInitialize}$

$\wedge \vee \wedge \textit{transaction}[i].\textit{state} = \textit{TransactionInProgress}$

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

$\wedge \neg \exists j \in \text{DOMAIN } \textit{transaction} :$

$\quad \wedge j < i$

$\quad \wedge \textit{transaction}[j].\textit{phase} = \textit{TransactionInitialize}$

$\quad \wedge \textit{transaction}[j].\textit{state} = \textit{TransactionInProgress}$

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

$\wedge \vee \wedge \text{DOMAIN } \textit{transaction}[i].\textit{targets} = \{ \}$

If the transaction is a change, the targets are taken

from the change values.

$$\begin{aligned}
& \wedge \vee \wedge \text{transaction}[i].\text{type} = \text{TransactionChange} \\
& \wedge \text{proposal}' = [t \in \text{DOMAIN } \text{proposal} \mapsto \\
& \quad \text{IF } t \in \text{DOMAIN } \text{transaction}[i].\text{change} \text{ THEN} \\
& \quad \quad \text{Append}(\text{proposal}[t], [type \mapsto \text{ProposalChange}, \\
& \quad \quad \quad index \mapsto i, \\
& \quad \quad \quad change \mapsto \\
& \quad \quad \quad [index \mapsto i, \\
& \quad \quad \quad \quad values \mapsto \text{transaction}[i].\text{change}[t]], \\
& \quad \quad \quad \text{rollback} \mapsto \\
& \quad \quad \quad [index \mapsto 0], \\
& \quad \quad \quad \text{dependency} \mapsto [index \mapsto 0], \\
& \quad \quad \quad \text{phase} \mapsto \text{ProposalInitialize}, \\
& \quad \quad \quad \text{state} \mapsto \text{ProposalInProgress}]) \\
& \quad \text{ELSE} \\
& \quad \quad \text{proposal}[t]] \\
& \wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{targets} = \\
& \quad [t \in \text{DOMAIN } \text{transaction}[i].\text{change} \mapsto \text{Len}(\text{proposal}'[t])]] \\
& \text{If the transaction is a rollback, the targets affected are} \\
& \text{the targets of the change transaction being rolled back.} \\
& \vee \wedge \text{transaction}[i].\text{type} = \text{TransactionRollback} \\
& \quad \text{If the rollback index is a valid } \text{Change} \text{ transaction,} \\
& \quad \text{initialize proposals for all of the } \text{Change} \text{ targets.} \\
& \wedge \vee \wedge \text{transaction}[i].\text{rollback} \in \text{DOMAIN } \text{transaction} \\
& \quad \wedge \text{transaction}[\text{transaction}[i].\text{rollback}].\text{type} = \text{TransactionChange} \\
& \quad \wedge \text{proposal}' = [t \in \text{DOMAIN } \text{proposal} \mapsto \\
& \quad \quad \text{IF } t \in \text{DOMAIN } \text{transaction}[\text{transaction}[i].\text{rollback}].\text{change} \text{ THEN} \\
& \quad \quad \quad \text{Append}(\text{proposal}[t], [type \mapsto \text{ProposalRollback}, \\
& \quad \quad \quad \quad index \mapsto i, \\
& \quad \quad \quad \quad change \mapsto \\
& \quad \quad \quad \quad [index \mapsto 0], \\
& \quad \quad \quad \quad \text{rollback} \mapsto \\
& \quad \quad \quad \quad [index \mapsto \text{transaction}[i].\text{rollback}], \\
& \quad \quad \quad \quad \text{dependency} \mapsto [index \mapsto 0], \\
& \quad \quad \quad \quad \text{phase} \mapsto \text{ProposalInitialize}, \\
& \quad \quad \quad \quad \text{state} \mapsto \text{ProposalInProgress}]) \\
& \quad \quad \text{ELSE} \\
& \quad \quad \quad \text{proposal}[t]] \\
& \quad \wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{targets} = \\
& \quad \quad [t \in \text{DOMAIN } \text{transaction}[\text{transaction}[i].\text{rollback}].\text{change} \mapsto \\
& \quad \quad \quad \text{Len}(\text{proposal}'[t])]] \\
& \quad \text{If the rollback index is not a valid } \text{Change} \text{ transaction} \\
& \quad \text{fail the } \text{Rollback} \text{ transaction.} \\
& \vee \wedge \vee \wedge \text{transaction}[i].\text{rollback} \in \text{DOMAIN } \text{transaction} \\
& \quad \wedge \text{transaction}[\text{transaction}[i].\text{rollback}].\text{type} = \text{TransactionRollback}
\end{aligned}$$

$\vee \text{transaction}[i].\text{rollback} \notin \text{DOMAIN } \text{transaction}$
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{state} = \text{TransactionFailed}]$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

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

$\vee \wedge \text{DOMAIN } \text{transaction}[i].\text{targets} \neq \{\}$
 If all proposals have been *Complete*, mark the transaction *Complete*.
 $\wedge \vee \wedge \forall t \in \text{DOMAIN } \text{transaction}[i].\text{targets} :$
 $\text{LET } p \triangleq \text{transaction}[i].\text{targets}[t]$
 IN
 $\wedge \text{proposal}[t][p].\text{phase} = \text{ProposalInitialize}$
 $\wedge \text{proposal}[t][p].\text{state} = \text{ProposalComplete}$
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{state} = \text{TransactionComplete}]$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

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

$\vee \wedge \exists t \in \text{DOMAIN } \text{transaction}[i].\text{targets} :$
 $\text{LET } p \triangleq \text{transaction}[i].\text{targets}[t]$
 IN
 $\wedge \text{proposal}[t][p].\text{phase} = \text{ProposalInitialize}$
 $\wedge \text{proposal}[t][p].\text{state} = \text{ProposalFailed}$
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{state} = \text{TransactionFailed}]$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

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.

$\vee \wedge \text{transaction}[i].\text{state} = \text{TransactionComplete}$
 $\wedge \forall t \in \text{DOMAIN } \text{transaction}[i].\text{targets} :$
 $\text{LET } p \triangleq \text{transaction}[i].\text{targets}[t]$
 IN
 $\wedge \text{proposal}[t][p].\text{dependency.index} \in \text{DOMAIN } \text{transaction}$
 $\wedge \text{transaction}[\text{proposal}[t][p].\text{dependency.index}].\text{isolation} = \text{Serializable}$
 $\Rightarrow \text{transaction}[\text{proposal}[t][p].\text{dependency.index}].\text{status}$
 $\in \{ \text{TransactionValidated}, \text{TransactionCommitted}, \text{TransactionApplied}, \text{TransactionFailed} \}$
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{phase} = \text{TransactionValidate},$
 $\phantom{\wedge \text{transaction}' = [} ![i].\text{state} = \text{TransactionInProgress}]$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

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

$\vee \wedge \text{transaction}[i].\text{state} = \text{TransactionFailed}$
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{phase} = \text{TransactionAbort},$
 $\phantom{\wedge \text{transaction}' = [} ![i].\text{state} = \text{TransactionInProgress}]$
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

$\vee \wedge \text{transaction}[i].\text{phase} = \text{TransactionValidate}$
 $\wedge \vee \wedge \text{transaction}[i].\text{state} = \text{TransactionInProgress}$

Move the transaction's proposals to the Validating state

$$\begin{aligned} & \wedge \vee \wedge \exists t \in \text{DOMAIN } \text{transaction}[i].\text{targets} : \\ & \quad \text{LET } p \triangleq \text{transaction}[i].\text{targets}[t] \\ & \quad \text{IN} \\ & \quad \wedge \text{proposal}[t][p].\text{phase} \neq \text{ProposalValidate} \\ & \quad \wedge \text{proposal}' = [\text{proposal EXCEPT } ![t] = \\ & \quad \quad [\text{proposal}[t] \text{ EXCEPT } ![p].\text{phase} = \text{ProposalValidate}, \\ & \quad \quad \quad ![p].\text{state} = \text{ProposalInProgress}]] \\ & \wedge \text{UNCHANGED } \langle \text{transaction} \rangle \end{aligned}$$

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

$$\begin{aligned} & \vee \wedge \forall t \in \text{DOMAIN } \text{transaction}[i].\text{targets} : \\ & \quad \text{LET } p \triangleq \text{transaction}[i].\text{targets}[t] \\ & \quad \text{IN} \\ & \quad \wedge \text{proposal}[t][p].\text{phase} = \text{ProposalValidate} \\ & \quad \wedge \text{proposal}[t][p].\text{state} = \text{ProposalComplete} \\ & \wedge \text{transaction}' = [\text{transaction EXCEPT } ![i].\text{state} = \text{TransactionComplete}, \\ & \quad \quad \quad ![i].\text{status} = \text{TransactionValidated}] \\ & \wedge \text{UNCHANGED } \langle \text{proposal} \rangle \end{aligned}$$

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

$$\begin{aligned} & \vee \wedge \exists t \in \text{DOMAIN } \text{transaction}[i].\text{targets} : \\ & \quad \text{LET } p \triangleq \text{transaction}[i].\text{targets}[t] \\ & \quad \text{IN} \\ & \quad \wedge \text{proposal}[t][p].\text{phase} = \text{ProposalValidate} \\ & \quad \wedge \text{proposal}[t][p].\text{state} = \text{ProposalFailed} \\ & \wedge \text{transaction}' = [\text{transaction EXCEPT } ![i].\text{state} = \text{TransactionFailed}] \\ & \wedge \text{UNCHANGED } \langle \text{proposal} \rangle \end{aligned}$$

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.

$$\begin{aligned} & \wedge \text{transaction}[i].\text{state} = \text{TransactionComplete} \\ & \wedge \forall t \in \text{DOMAIN } \text{transaction}[i].\text{targets} : \\ & \quad \text{LET } p \triangleq \text{transaction}[i].\text{targets}[t] \\ & \quad \text{IN} \\ & \quad \wedge \text{proposal}[t][p].\text{dependency.index} \in \text{DOMAIN } \text{transaction} \\ & \quad \wedge \text{transaction}[\text{proposal}[t][p].\text{dependency.index}].\text{isolation} = \text{Serializable} \\ & \quad \Rightarrow \text{transaction}[\text{proposal}[t][p].\text{dependency.index}].\text{status} \\ & \quad \quad \in \{ \text{TransactionCommitted}, \text{TransactionApplied}, \text{TransactionAborted} \} \\ & \wedge \text{transaction}' = [\text{transaction EXCEPT } ![i].\text{phase} = \text{TransactionCommit}, \\ & \quad \quad \quad ![i].\text{state} = \text{TransactionInProgress}] \\ & \wedge \text{UNCHANGED } \langle \text{proposal} \rangle \end{aligned}$$

If the transaction failed validation, proceed to the *Abort* phase

To ensure indexes are still updated for the target configurations.

$$\begin{aligned} & \wedge \text{transaction}[i].\text{state} = \text{TransactionFailed} \\ & \wedge \text{transaction}' = [\text{transaction EXCEPT } ![i].\text{phase} = \text{TransactionAbort}, \end{aligned}$$

$$\begin{aligned} & \wedge \vee \wedge \exists t \in \text{DOMAIN } transaction[i].targets : \\ & \quad \text{LET } p \triangleq transaction[i].targets[t] \\ & \quad \text{IN} \\ & \quad \wedge proposal[t][p].phase \neq ProposalValidate \\ & \quad \wedge proposal' = [proposal \text{ EXCEPT } ![t] = \\ & \hspace{8em} [proposal[t] \text{ EXCEPT } ![p].phase = ProposalValidate, \\ & \hspace{8em} ![p].state = ProposalInProgress]] \end{aligned}$$

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

[illegible]

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

$$\begin{array}{l} \vee \wedge \exists t \in \text{DOMAIN } \text{transaction}[i].\text{targets} : \\ \quad \text{LET } p \triangleq \text{transaction}[i].\text{targets}[t] \\ \quad \text{IN} \\ \quad \quad \wedge \text{proposal}[t][p].\text{phase} = \text{ProposalValidate} \\ \quad \quad \wedge \text{proposal}[t][p].\text{state} = \text{ProposalFailed} \\ \wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{state} = \text{TransactionFailed}] \\ \wedge \text{UNCHANGED } \langle \text{proposal} \rangle \end{array}$$

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.

[illegible]

to ensure indexes are still updated for the target configurations.

$$\begin{aligned} & \vee \wedge transaction[i].state = TransactionFailed \\ & \wedge transaction' = [transaction \text{ EXCEPT } ![i].phase = TransactionAbort, \end{aligned}$$

$![i].state = TransactionInProgress]$

\wedge UNCHANGED $\langle proposal \rangle$

$\vee \wedge transaction[i].phase = TransactionCommit$

$\wedge \vee \wedge transaction[i].state = TransactionInProgress$

Move the transaction's proposals to the Committing state

$\wedge \vee \wedge \exists t \in \text{DOMAIN } transaction[i].targets :$

$\text{LET } p \triangleq transaction[i].targets[t]$

IN

$\wedge proposal[t][p].phase \neq ProposalCommit$

$\wedge proposal' = [proposal \text{ EXCEPT } ![t] =$

$[proposal[t] \text{ EXCEPT } ![p].phase = ProposalCommit,$

$![p].state = ProposalInProgress]]$

\wedge UNCHANGED $\langle transaction \rangle$

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

$\vee \wedge \forall t \in \text{DOMAIN } transaction[i].targets :$

$\text{LET } p \triangleq transaction[i].targets[t]$

IN

$\wedge proposal[t][p].phase = ProposalCommit$

$\wedge proposal[t][p].state = ProposalComplete$

$\wedge transaction' = [transaction \text{ EXCEPT } ![i].state = TransactionComplete,$

$![i].status = TransactionCommitted]$

\wedge 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.

$\vee \wedge transaction[i].state = TransactionComplete$

$\wedge \forall t \in \text{DOMAIN } transaction[i].targets :$

$\text{LET } p \triangleq transaction[i].targets[t]$

IN

$\wedge proposal[t][p].dependency.index \in \text{DOMAIN } transaction$

$\wedge transaction[proposal[t][p].dependency.index].isolation = Serializable$

$\Rightarrow transaction[proposal[t][p].dependency.index].status$

$\in \{TransactionApplied, TransactionAborted\}$

$\wedge transaction' = [transaction \text{ EXCEPT } ![i].phase = TransactionApply,$

$![i].state = TransactionInProgress]$

\wedge UNCHANGED $\langle proposal \rangle$

$\vee \wedge transaction[i].phase = TransactionApply$

$\wedge transaction[i].state = TransactionInProgress$

Move the transaction's proposals to the Applying state

$\wedge \vee \wedge \exists t \in \text{DOMAIN } transaction[i].targets :$

$\text{LET } p \triangleq transaction[i].targets[t]$

IN

$\wedge proposal[t][p].phase \neq ProposalApply$

$\wedge proposal' = [proposal \text{ EXCEPT } ![t] =$

[proposal[t] EXCEPT ![p].phase = ProposalApply,
![p].state = ProposalInProgress]]

∧ UNCHANGED ⟨transaction⟩

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

∨ ∧ ∀ t ∈ DOMAIN transaction[i].targets :

LET p ≜ transaction[i].targets[t]

IN

∧ proposal[t][p].phase = ProposalApply

∧ proposal[t][p].state = ProposalComplete

∧ transaction' = [transaction EXCEPT ![i].state = TransactionComplete,
![i].status = TransactionApplied]

∧ UNCHANGED ⟨proposal⟩

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

∨ ∧ ∃ t ∈ DOMAIN transaction[i].targets :

LET p ≜ transaction[i].targets[t]

IN

∧ proposal[t][p].phase = ProposalApply

∧ proposal[t][p].state = ProposalFailed

∧ transaction' = [transaction EXCEPT ![i].state = TransactionFailed]

∧ UNCHANGED ⟨proposal⟩

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

∨ ∧ transaction[i].phase = TransactionAbort

∧ transaction[i].state = TransactionInProgress

Move the transaction's proposals to the Aborting state

∧ ∨ ∧ ∃ t ∈ DOMAIN transaction[i].targets :

LET p ≜ transaction[i].targets[t]

IN

∧ proposal[t][p].phase ≠ ProposalAbort

∧ proposal' = [proposal EXCEPT ![t] =
[proposal[t] EXCEPT ![p].phase = ProposalAbort,
![p].state = ProposalInProgress]]

∧ UNCHANGED ⟨transaction⟩

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

∨ ∧ ∀ t ∈ DOMAIN transaction[i].targets :

LET p ≜ transaction[i].targets[t]

IN

∧ proposal[t][p].phase = ProposalAbort

∧ proposal[t][p].state = ProposalComplete

∧ transaction' = [transaction EXCEPT ![i].state = TransactionComplete,
![i].status = TransactionAborted]

∧ UNCHANGED ⟨proposal⟩

Formal specification, constraints, and theorems.

$$\begin{aligned} \textit{InitTransaction} &\triangleq \\ &\wedge \textit{transaction} = [i \in \{\} \mapsto \\ &\quad [type \mapsto \textit{TransactionChange}, \\ &\quad \quad phase \mapsto \textit{TransactionInitialize}, \\ &\quad \quad state \mapsto \textit{TransactionInProgress}, \\ &\quad \quad status \mapsto \textit{TransactionPending}]] \\ &\wedge \textit{Trace!Init} \\ \textit{NextTransaction} &\triangleq \\ &\vee \exists i \in \text{DOMAIN } \textit{transaction} : \\ &\quad \textit{Trace!Step}(\text{"Reconcile"}, \textit{ReconcileTransaction}(i), [\textit{index} \mapsto i]) \end{aligned}$$

\ * Modification History
\ * Last modified Sun Feb 20 10:04:00 PST 2022 by jordanhalterman
\ * Created Sun Feb 20 02:20:45 PST 2022 by jordanhalterman