

---

MODULE *Transaction*

---

INSTANCE *Naturals*  
 INSTANCE *FiniteSets*  
 INSTANCE *Sequences*  
 INSTANCE *TLC*

---

An empty constant  
 CONSTANT *Nil*

Transaction type constants  
 CONSTANTS  
     *Change*,  
     *Rollback*

$Type \triangleq \{Change, Rollback\}$

Proposal phase constants  
 CONSTANTS  
     *Commit*,  
     *Apply*

Status constants  
 CONSTANTS  
     *Pending*,  
     *InProgress*,  
     *Complete*,  
     *Aborted*,  
     *Failed*

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

$Done \triangleq \{Complete, Aborted, Failed\}$

The set of all nodes  
 CONSTANT *Node*

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

---

Variables defined by other modules.  
 VARIABLES  
     *proposal*,  
     *configuration*,

*mastership*,  
*target*,  
*history*

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

VARIABLE *transaction*

$TypeOK \triangleq$   
 $\forall i \in \text{DOMAIN } transaction :$   
 $\wedge transaction[i].type \in Type$   
 $\wedge transaction[i].index \in Nat$   
 $\wedge transaction[i].init \in Status$   
 $\wedge transaction[i].commit \in Status$   
 $\wedge transaction[i].apply \in Status$   
 $\wedge \forall p \in \text{DOMAIN } transaction[i].values :$   
 $transaction[i].values[p] \neq Nil \Rightarrow transaction[i].values[p] \in \text{STRING}$

$Test \triangleq$  INSTANCE *Test* WITH  
*File*  $\leftarrow$  "Transaction.log",  
*CurrState*  $\leftarrow$  [  
*transactions*  $\mapsto transaction$ ,  
*proposals*  $\mapsto proposal$ ,  
*configuration*  $\mapsto configuration$ ,  
*mastership*  $\mapsto mastership$ ,  
*target*  $\mapsto target$ ],  
*SuccState*  $\leftarrow$  [  
*transactions*  $\mapsto transaction'$ ,  
*proposals*  $\mapsto proposal'$ ,  
*configuration*  $\mapsto configuration'$ ,  
*mastership*  $\mapsto mastership'$ ,  
*target*  $\mapsto target'$ ]

---

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

$RequestChange(p, v) \triangleq$   
 $\wedge transaction' = Append(transaction, [$   
 $type \mapsto Change,$   
 $index \mapsto 0,$   
 $values \mapsto (p :> v),$   
 $init \mapsto InProgress,$   
 $commit \mapsto Pending,$   
 $apply \mapsto Pending])$

$\wedge \text{UNCHANGED } \langle \text{proposal}, \text{configuration}, \text{mastership}, \text{target}, \text{history} \rangle$

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

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

$\wedge \text{transaction}' = \text{Append}(\text{transaction}, [$   
 $\quad \text{type} \mapsto \text{Rollback},$   
 $\quad \text{index} \mapsto i,$   
 $\quad \text{values} \mapsto \text{Empty},$   
 $\quad \text{init} \mapsto \text{InProgress},$   
 $\quad \text{commit} \mapsto \text{Pending},$   
 $\quad \text{apply} \mapsto \text{Pending}])$

$\wedge \text{UNCHANGED } \langle \text{proposal}, \text{configuration}, \text{mastership}, \text{target}, \text{history} \rangle$

---

This section models the *Transaction* log reconciler.

LOCAL  $\text{IsInitialized}(i) \triangleq$

$i \in \text{DOMAIN } \text{transaction} \Rightarrow \text{transaction}[i].\text{init} \in \text{Done}$

LOCAL  $\text{IsCommitted}(i) \triangleq$

$i \in \text{DOMAIN } \text{transaction} \Rightarrow \text{transaction}[i].\text{commit} \in \text{Done}$

LOCAL  $\text{IsApplied}(i) \triangleq$

$i \in \text{DOMAIN } \text{transaction} \Rightarrow \text{transaction}[i].\text{apply} \in \text{Done}$

$\text{InitChange}(n, i) \triangleq$

$\wedge \vee \wedge \text{transaction}[i].\text{init} = \text{InProgress}$

If the prior transaction has been initialized, initialize the transaction by  
appending the proposal and updating the transaction index.

$\wedge \text{IsInitialized}(i - 1)$

$\wedge \text{proposal}' = \text{Append}(\text{proposal}, [$   
 $\quad \text{change} \mapsto [$   
 $\quad \quad \text{phase} \mapsto \text{Commit},$   
 $\quad \quad \text{state} \mapsto \text{Pending},$   
 $\quad \quad \text{values} \mapsto [$   
 $\quad \quad \quad p \in \text{DOMAIN } \text{transaction}[i].\text{values} \mapsto [$   
 $\quad \quad \quad \quad \text{index} \mapsto \text{Len}(\text{proposal}) + 1,$   
 $\quad \quad \quad \quad \text{value} \mapsto \text{transaction}[i].\text{values}[p]]],$   
 $\quad \text{rollback} \mapsto [$   
 $\quad \quad \text{phase} \mapsto \text{Nil},$   
 $\quad \quad \text{state} \mapsto \text{Nil},$   
 $\quad \quad \text{revision} \mapsto 0,$   
 $\quad \quad \text{values} \mapsto \text{Empty}])$

$\wedge \text{transaction}' = [\text{transaction } \text{EXCEPT } ![i].\text{index} = \text{Len}(\text{proposal}'),$   
 $\quad \quad \quad ![i].\text{init} = \text{Complete}]$



If the prior change failed being applied, it must be rolled back before new changes can be applied.  
 $\wedge \wedge \text{transaction}[i].\text{index} - 1 \in \text{DOMAIN } \text{proposal}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{change.phase} = \text{Apply}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{change.state} = \text{Failed}$   
 $\Rightarrow \wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{rollback.phase} = \text{Apply}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{rollback.state} = \text{Complete}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{InProgress}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$   
 If the commit phase was aborted or failed, abort the apply phase once the previous transaction has completed the apply phase.  
 $\vee \wedge \text{transaction}[i].\text{commit} \in \{\text{Aborted}, \text{Failed}\}$   
 A transaction cannot be applied until the prior transaction has been applied.  
 $\wedge \text{IsApplied}(i - 1)$   
 If the prior change failed being applied, it must be rolled back before new changes can be applied.  
 $\wedge \wedge \text{transaction}[i].\text{index} - 1 \in \text{DOMAIN } \text{proposal}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{change.phase} = \text{Apply}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{change.state} = \text{Failed}$   
 $\Rightarrow \wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{rollback.phase} = \text{Apply}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index} - 1].\text{rollback.state} = \text{Complete}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{Aborted}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$   
 $\vee \wedge \text{transaction}[i].\text{apply} = \text{InProgress}$   
 $\wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{change.phase} = \text{Apply}$   
 If the change apply is still in the *Pending* state, set it to *InProgress*.  
 $\wedge \vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{change.state} = \text{Pending}$   
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![\text{transaction}[i].\text{index}].\text{change.state} = \text{InProgress}]$   
 $\wedge \text{UNCHANGED } \langle \text{transaction} \rangle$   
 If the change apply is *Complete*, mark the transaction *Complete*.  
 $\vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{change.state} = \text{Complete}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{Complete}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$   
 If the change apply *Failed*, mark the transaction *Failed*.  
 $\vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{change.state} = \text{Failed}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{Failed}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

$\text{ReconcileChange}(n, i) \triangleq$   
 $\wedge \text{transaction}[i].\text{type} = \text{Change}$   
 $\wedge \vee \text{InitChange}(n, i)$   
 $\vee \text{CommitChange}(n, i)$   
 $\vee \text{ApplyChange}(n, i)$

$InitRollback(n, i) \triangleq$   
 $\wedge \vee \wedge transaction[i].init = InProgress$   
 Rollbacks cannot be initialized until all prior transactions have been initialized.  
 $\wedge IsInitialized(i - 1)$   
 Rollback transactions must target valid proposal index.  
 $\wedge \vee \wedge transaction[i].index \in DOMAIN\ proposal$   
 To roll back a transaction, all subsequent transactions must be rolled back first.  
 Check whether the following proposal is being rolled back.  
 $\wedge \vee \wedge transaction[i].index + 1 \in DOMAIN\ proposal \Rightarrow$   
 $proposal[transaction[i].index + 1].rollback.phase \neq Nil$   
 $\wedge transaction' = [transaction\ EXCEPT\ ![i].init = Complete]$   
 If the subsequent proposal is not being rolled back, fail the rollback transaction.  
 $\vee \wedge transaction[i].index + 1 \in DOMAIN\ proposal$   
 $\wedge proposal[transaction[i].index + 1].rollback.phase = Nil$   
 $\wedge transaction' = [transaction\ EXCEPT\ ![i].init = Failed]$   
 If the rollback index is not a valid proposal index, fail the rollback request.  
 $\vee \wedge transaction[i].index \notin DOMAIN\ proposal$   
 $\wedge transaction' = [transaction\ EXCEPT\ ![i].init = Failed]$   
 $\wedge UNCHANGED\ \langle proposal \rangle$

$CommitRollback(n, i) \triangleq$   
 $\wedge \vee \wedge transaction[i].commit = Pending$   
 A transaction cannot be committed until the prior transaction has been committed.  
 In the case of rollbacks, we serialize all state changes to ensure consistency  
 when rolling back changes.  
 $\wedge IsCommitted(i - 1)$   
 If the transaction was initialized successfully, commit the rollback.  
 $\wedge \vee \wedge transaction[i].init = Complete$   
 If the target proposal is not yet being rolled back, transition the proposal.  
 $\wedge \vee \wedge proposal[transaction[i].index].rollback.phase = Nil$   
 Update the proposal's rollback state based on its change state.  
 $\wedge \vee \wedge proposal[transaction[i].index].change.phase = Commit$   
 If the target change is still pending, abort the change and rollback.  
 $\wedge \vee \wedge proposal[transaction[i].index].change.state = Pending$   
 $\wedge proposal' = [proposal\ EXCEPT\ ![transaction[i].index].change.state = Aborted,$   
 $![transaction[i].index].rollback.phase = Commit$   
 $![transaction[i].index].rollback.state = Aborted]$   
 $\wedge UNCHANGED\ \langle transaction \rangle$   
 If the target change is complete, start the rollback commit phase.  
 $\vee \wedge proposal[transaction[i].index].change.state = Complete$   
 $\wedge proposal' = [proposal\ EXCEPT\ ![transaction[i].index].rollback.phase = Commit,$   
 $![transaction[i].index].rollback.state = Pending]$   
 $\wedge UNCHANGED\ \langle transaction \rangle$   
 If the target change failed commit, complete the rollback commit.



$\wedge IsApplied(i - 1)$   
 If the commit phase was completed successfully, start the apply phase.  
 $\wedge \vee \wedge transaction[i].commit = Complete$   
 If the target rollback is not yet being applied, transition the rollback.  
 $\wedge \vee \wedge proposal[transaction[i].index].rollback.phase = Commit$   
 Update the proposal's rollback state based on its change state.  
 $\wedge \vee \wedge proposal[transaction[i].index].change.phase = Apply$   
 If the target change is still pending, abort the change and rollback.  
 $\wedge \vee \wedge proposal[transaction[i].index].change.state = Pending$   
 $\wedge proposal' = [proposal \text{ EXCEPT } ![transaction[i].index].change.state = Aborted,$   
 $![transaction[i].index].rollback.phase = Apply,$   
 $![transaction[i].index].rollback.state = Aborted]$   
 $\wedge \text{UNCHANGED } \langle transaction \rangle$   
 If the target change is complete, start the rollback apply phase.  
 $\vee \wedge proposal[transaction[i].index].change.state = Complete$   
 $\wedge proposal' = [proposal \text{ EXCEPT } ![transaction[i].index].rollback.phase = Apply,$   
 $![transaction[i].index].rollback.state = Pending]$   
 $\wedge \text{UNCHANGED } \langle transaction \rangle$   
 If the target change failed apply, complete the rollback apply.  
 $\vee \wedge proposal[transaction[i].index].change.state \in \{Aborted, Failed\}$   
 $\wedge transaction' = [transaction \text{ EXCEPT } ![i].apply = Complete]$   
 $\wedge \text{UNCHANGED } \langle proposal \rangle$   
 If the target change is in the *Commit* phase, abort the change and rollback.  
 $\vee \wedge proposal[transaction[i].index].change.phase = Commit$   
 $\wedge proposal' = [proposal \text{ EXCEPT } ![transaction[i].index].change.state = Aborted,$   
 $![transaction[i].index].rollback.phase = Apply,$   
 $![transaction[i].index].rollback.state = Aborted]$   
 $\wedge \text{UNCHANGED } \langle transaction \rangle$   
 If the target rollback is being applied, transition the underlying proposal.  
 $\vee \wedge proposal[transaction[i].index].rollback.phase = Apply$   
 If the target proposal is being rolled back, begin the rollback apply  
 once the prior transaction has completed the apply phase.  
 $\wedge \vee \wedge proposal[transaction[i].index].rollback.state = Pending$   
 $\wedge transaction' = [transaction \text{ EXCEPT } ![i].apply = InProgress]$   
 $\wedge \text{UNCHANGED } \langle proposal \rangle$   
 If the target rollback was aborted, abort the transaction rollback  
 once the prior transaction has completed the apply phase.  
 $\vee \wedge proposal[transaction[i].index].rollback.state = Aborted$   
 $\wedge transaction' = [transaction \text{ EXCEPT } ![i].apply = Aborted]$   
 $\wedge \text{UNCHANGED } \langle proposal \rangle$   
 If the transaction failed initialization, abort the apply phase.  
 $\vee \wedge transaction[i].init \in \{Aborted, Failed\}$   
 $\wedge transaction' = [transaction \text{ EXCEPT } ![i].apply = Aborted]$   
 $\wedge \text{UNCHANGED } \langle proposal \rangle$   
 $\vee \wedge transaction[i].apply = InProgress$



$\wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{rollback}.\text{phase} = \text{Apply}$   
 If the rollback apply is still in the *Pending* state, set it to *InProgress*.  
 $\wedge \vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{rollback}.\text{state} = \text{Pending}$   
 $\wedge \text{proposal}' = [\text{proposal} \text{ EXCEPT } ![\text{transaction}[i].\text{index}].\text{rollback}.\text{state} = \text{InProgress}]$   
 $\wedge \text{UNCHANGED } \langle \text{transaction} \rangle$   
 If the rollback apply is *Complete*, mark the transaction *Complete*.  
 $\vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{rollback}.\text{state} = \text{Complete}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{Complete}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$   
 If the rollback apply *Failed*, mark the transaction *Failed*.  
 $\vee \wedge \text{proposal}[\text{transaction}[i].\text{index}].\text{rollback}.\text{state} = \text{Failed}$   
 $\wedge \text{transaction}' = [\text{transaction} \text{ EXCEPT } ![i].\text{apply} = \text{Failed}]$   
 $\wedge \text{UNCHANGED } \langle \text{proposal} \rangle$

$\text{ReconcileRollback}(n, i) \triangleq$   
 $\wedge \text{transaction}[i].\text{type} = \text{Rollback}$   
 $\wedge \vee \text{InitRollback}(n, i)$   
 $\vee \text{CommitRollback}(n, i)$   
 $\vee \text{ApplyRollback}(n, i)$

Reconcile a transaction  
 $\text{ReconcileTransaction}(n, i) \triangleq$   
 $\wedge i \in \text{DOMAIN } \text{transaction}$   
 $\wedge \vee \text{ReconcileChange}(n, i)$   
 $\vee \text{ReconcileRollback}(n, i)$   
 $\wedge \text{UNCHANGED } \langle \text{configuration}, \text{mastership}, \text{target}, \text{history} \rangle$

---