

---

MODULE *TwoPhaseLocking*

---

EXTENDS *FiniteSets*, *Naturals*, *Sequences*, *TLC*

CONSTANT *Proc*, *Object*

VARIABLE  
*transact*,  
*history*,  
*state*,  
*store*,  
*READ*,      read lock  
*WRITE*      write lock

*vars*  $\triangleq$   $\langle$   
*transact*,  
*history*,  
*state*,  
*store*,  
*READ*,  
*WRITE*  
 $\rangle$

Transaction is a set of all possible transactions

*Transaction*  $\triangleq$   
 LET *Op*  $\triangleq$   $[f : \{\text{"Read"}, \text{"Write"}\}, obj : Object]$   
       *seq*(*S*)  $\triangleq$  UNION  $\{[1 \dots n \rightarrow S] : n \in Nat\}$   
 IN     $\{Append(op, [f \mapsto \text{"Commit"}]) : op \in seq(Op)\}$

*Init*  $\triangleq$   
 $\wedge \exists tx \in [Proc \rightarrow Transaction] : transact = tx$   
 $\wedge history = \langle \rangle$   
 $\wedge state = [proc \in Proc \mapsto \text{"Init"}]$   
 $\wedge store = [obj \in Object \mapsto 0]$   
 $\wedge READ = [obj \in Object \mapsto \{\}]$   
 $\wedge WRITE = [obj \in Object \mapsto \{\}]$

*updateHistory*(*self*, *hd*, *tl*, *val*)  $\triangleq$   
 $\wedge history' = Append(history, [proc \mapsto self, op \mapsto hd, val \mapsto val])$   
 $\wedge transact' = [transact \text{ EXCEPT } ![self] = tl]$

*ReadLongDurationLock*(*self*, *hd*, *tl*)  $\triangleq$   
 $\wedge state[self] \in \{\text{"Init"}, \text{"Running"}\}$   
 $\wedge hd.f = \text{"Read"}$   
 $\wedge WRITE[hd.obj] \in \{\{\}, \{self\}\}$   
 $\wedge READ' = [READ \text{ EXCEPT } ![hd.obj] = READ[hd.obj] \cup \{self\}]$   
 $\wedge updateHistory(self, hd, tl, store[hd.obj])$   
 $\wedge \text{IF } state[self] = \text{"Init"}$

$$\begin{aligned}
& \text{THEN } \wedge state' = [state \text{ EXCEPT } ![self] = \text{"Running"}] \\
& \quad \wedge \text{UNCHANGED } \langle store, WRITE \rangle \\
& \text{ELSE UNCHANGED } \langle state, store, WRITE \rangle \\
\\
ReadShortDurationLock(self, hd, tl) & \triangleq \\
& \wedge state[self] \in \{\text{"Init"}, \text{"Running"}\} \\
& \wedge hd.f = \text{"Read"} \\
& \wedge WRITE[hd.obj] \in \{\{\}, \{self\}\} \\
& \wedge updateHistory(self, hd, tl, store[hd.obj]) \\
& \wedge \text{IF } state[self] = \text{"Init"} \\
& \quad \text{THEN } \wedge state' = [state \text{ EXCEPT } ![self] = \text{"Running"}] \\
& \quad \quad \wedge \text{UNCHANGED } \langle store, READ, WRITE \rangle \\
& \quad \text{ELSE UNCHANGED } \langle state, store, READ, WRITE \rangle \\
\\
Read(self, hd, tl) & \triangleq ReadLongDurationLock(self, hd, tl) \\
\\
Write(self, hd, tl) & \triangleq \\
& \wedge state[self] \in \{\text{"Init"}, \text{"Running"}\} \\
& \wedge hd.f = \text{"Write"} \\
& \wedge WRITE[hd.obj] \in \{\{\}, \{self\}\} \\
& \wedge WRITE' = [WRITE \text{ EXCEPT } ![hd.obj] = WRITE[hd.obj] \cup \{self\}] \\
& \wedge READ[hd.obj] \in \text{SUBSET } WRITE'[hd.obj] \\
& \wedge store' = [store \text{ EXCEPT } ![hd.obj] = store[hd.obj] + 1] \\
& \wedge updateHistory(self, hd, tl, store[hd.obj] + 1) \\
& \wedge \text{IF } state[self] = \text{"Init"} \\
& \quad \text{THEN } \wedge state' = [state \text{ EXCEPT } ![self] = \text{"Running"}] \\
& \quad \quad \wedge \text{UNCHANGED } \langle READ \rangle \\
& \quad \text{ELSE UNCHANGED } \langle state, READ \rangle \\
\\
Commit(self, hd, tl) & \triangleq \\
& \wedge state[self] \in \{\text{"Init"}, \text{"Running"}\} \\
& \wedge hd.f = \text{"Commit"} \\
& \wedge updateHistory(self, hd, tl, 0) \\
& \wedge state' = [state \text{ EXCEPT } ![self] = \text{"Commit"}] \\
& \wedge READ' = [obj \in Object \mapsto READ[obj] \setminus \{self\}] \\
& \wedge WRITE' = [obj \in Object \mapsto WRITE[obj] \setminus \{self\}] \\
& \wedge \text{UNCHANGED } \langle store \rangle \\
\\
Next & \triangleq \exists self \in Proc \\
& : \wedge transact[self] \neq \langle \rangle \\
& \quad \wedge \text{LET } hd \triangleq Head(transact[self]) \\
& \quad \quad tl \triangleq Tail(transact[self]) \\
& \quad \text{IN } \vee Read(self, hd, tl) \\
& \quad \quad \vee Write(self, hd, tl) \\
& \quad \quad \vee Commit(self, hd, tl) \\
\\
Spec & \triangleq Init \wedge \Box [Next]_{vars} \wedge WF_{vars}(Next)
\end{aligned}$$

$Invariants \triangleq$   
 $\wedge \forall proc \in Proc$   
 $\quad : state[proc] \in \{ "Init", "Running", "Commit" \}$   
 $\wedge \forall obj \in Object$   
 $\quad : Cardinality(WRITE[obj]) \in \{0, 1\}$   
 $\wedge \forall obj \in Object$   
 $\quad : Cardinality(WRITE[obj]) \neq 0 \Rightarrow READ[obj] \in SUBSET WRITE[obj]$

Serializable tests if a history is serializable

RECURSIVE  $consistent(-, -)$   
 $consistent(s, hist) \triangleq$   
 IF  $hist = \langle \rangle$   
 THEN TRUE  
 ELSE LET  $hd \triangleq Head(hist)$   
     IN CASE  $hd.op.f = "Read"$   
          $\rightarrow s[hd.op.obj] = hd.val \wedge consistent(s, Tail(hist))$   
     □  $hd.op.f = "Write"$   
          $\rightarrow consistent([s \text{ EXCEPT } ![hd.op.obj] = hd.val], Tail(hist))$   
     □ OTHER  
          $\rightarrow consistent(s, Tail(hist))$

$Serializable \triangleq$   
 LET  $Tx \triangleq \{ SelectSeq(history, LAMBDA x : x.proc = proc) : proc$   
      $\in \{ proc \in Proc : state[proc] = "Commit" \} \}$   
      $perms \triangleq \{ f \in [1 .. Cardinality(Tx) \rightarrow Tx]$   
          $: \forall tx \in Tx$   
              $: \exists proc \in 1 .. Cardinality(Tx) : f[proc] = tx \}$   
 IN LET RECURSIVE  $concat(-, -, -, -)$   
      $concat(f, n, size, acc) \triangleq$   
         IF  $n > size$  THEN  $acc$  ELSE  $concat(f, n + 1, size, acc \circ f[n])$   
     IN  $\exists perm \in perms$   
          $: consistent([obj \in Object \mapsto 0],$   
              $concat(perm, 1, Cardinality(Tx), \langle \rangle))$   
          $\wedge \vee Cardinality(Tx) < 2$   
          $\vee PrintT(\langle history, concat(perm, 1, Cardinality(Tx), \langle \rangle) \rangle)$

\*\*\*\*\*  
 Properties assert that if some of transactions successfully commit then the history of the committed transactions is serializable  
 \*\*\*\*\*

$Properties \triangleq \Box Serializable$

EventuallyAllCommit is used to detect a deadlock

$EventuallyAllCommit \triangleq \Diamond \Box (\exists proc \in Proc : state[proc] = "Commit")$

THEOREM  $Spec \Rightarrow \Box Invariants \wedge Properties$

---

\\* Modification History  
\\* Last modified Sat *Feb* 17 13:58:11 *JST* 2018 by *takayuki*  
\\* Created Sat *Feb* 17 10:34:44 *JST* 2018 by *takayuki*