PODC 2000 TLA$^+$ Tutorial

Leslie Lamport

16 July 2000

MODULE *Mutex*

CONSTANT *Proc*

VARIABLE *state*

$$Init \triangleq state = [p \in Proc \mapsto \text{“idle”}]$$

$$Request(p) \triangleq \land\ state[p] = \text{“idle”} \\ \land\ state' = [state \text{ EXCEPT } ![p] = \text{“waiting”}]$$

$$Acquire(p) \triangleq \land\ state[p] = \text{“waiting”} \\ \land\ \forall\, q \in Proc : state[q] \neq \text{“owner”} \\ \land\ state' = [state \text{ EXCEPT } ![p] = \text{“owner”}]$$

$$Release(p) \triangleq \land\ state[p] = \text{“owner”} \\ \land\ state' = [state \text{ EXCEPT } ![p] = \text{“idle”}]$$

$$Next \triangleq \exists\, p \in Proc : Request(p) \lor Acquire(p) \lor Release(p)$$

$$StarvationFree \triangleq \forall\, p \in Proc : \mathrm{SF}_{state}(Acquire(p))$$

$$Spec \triangleq Init \land \Box[Next]_{state} \land StarvationFree$$

```
------------------------ MODULE Mutex ------------------------
CONSTANT Proc
VARIABLE state
------------------------------------------------------------
Init == state = [p \in Proc |-> "idle"]

Request(p) == /\ state[p] = "idle"
              /\ state' = [state EXCEPT ![p] = "waiting"]

Acquire(p) == /\ state[p] = "waiting"
              /\ \A q \in Proc : state[q] # "owner"
              /\ state' = [state EXCEPT ![p] = "owner"]

Release(p) == /\ state[p] = "owner"
              /\ state' = [state EXCEPT ![p] = "idle"]

Next == \E p \in Proc : Request(p) \/ Acquire(p) \/ Release(p)
------------------------------------------------------------
StarvationFree == \A p \in Proc : SF_state(Acquire(p))

Spec == Init /\ [][Next]_state /\ StarvationFree
============================================================
```

MODULE *TimeClocks*

EXTENDS *Naturals*, *Sequences*

CONSTANT $Proc$, $\_ \ll \_$

ASSUME $\forall\, p \in Proc : \land \neg p \ll p$
$\land \forall\, q \in Proc \setminus \{p\} : (p \ll q) \lor (q \ll p)$
$\land \forall\, q, r \in Proc : (p \ll q) \land (q \ll r) \Rightarrow (p \ll r)$

$a \prec b \triangleq \lor\ a.TS < b.TS$
$\lor\ (a.TS = b.TS) \land (a.proc \ll b.proc)$

VARIABLES $state$, $msgQ$, $reqSet$ $clock$, $lastTSent$, $lastTRcvd$

$vars \triangleq \langle state, msgQ, reqSet, clock, lastTSent, lastTRcvd \rangle$

$Init \triangleq \land\ state = [p \in Proc \mapsto \text{“idle”}]$
$\land\ msgQ = [p \in Proc \mapsto [q \in Proc \setminus \{p\} \mapsto \langle\rangle]]$
$\land\ reqSet = [p \in Proc \mapsto \{\}]$
$\land\ clock \in [Proc \to Nat]$
$\land\ lastTSent = [p \in Proc \mapsto [q \in Proc \setminus \{p\} \mapsto 0]]$
$\land\ lastTRcvd = [p \in Proc \mapsto [q \in Proc \setminus \{p\} \mapsto 0]]$

$Request(p) \triangleq$
$\land\ state[p] = \text{“idle”}$
$\land\ state' = [state \text{ EXCEPT } ![p] = \text{“waiting”}]$
$\land\ \exists\, n \in Nat :$
$\land\ clock' = [clock \text{ EXCEPT } ![p] = n]$
$\land\ n > clock[p]$
$\land$ LET $msg \triangleq [TS \mapsto n, proc \mapsto p, cmd \mapsto \text{“acquire”}]$
IN $\land\ msgQ' = [msgQ \text{ EXCEPT } ![p] =$
$[q \in Proc \setminus \{p\} \mapsto Append(@[q], msg)]]$
$\land\ reqSet' = [reqSet \text{ EXCEPT } ![p] = @ \cup \{msg\}]$
$\land\ lastTSent' = [lastTSent \text{ EXCEPT } ![p] = [q \in Proc \setminus \{p\} \mapsto n]]$
$\land$ UNCHANGED $lastTRcvd$

$Acquire(p) \triangleq$
LET $pReq \triangleq$ CHOOSE $req \in reqSet[p] : req.proc = p$
IN $\land\ state[p] = \text{“waiting”}$
$\land\ \forall\, req \in reqSet[p] \setminus \{pReq\} : pReq \prec req$
$\land\ \forall\, q \in Proc \setminus \{p\} : pReq \prec [TS \mapsto lastTRcvd[p][q] + 1, proc \mapsto q]$
$\land\ state' = [state \text{ EXCEPT } ![p] = \text{“owner”}]$
$\land\ reqSet' = [reqSet \text{ EXCEPT } ![p] = @ \setminus \{pReq\}]$
$\land$ UNCHANGED $\langle msgQ, clock, lastTSent, lastTRcvd \rangle$

$$
\begin{array}{l}
Release(p) \triangleq \\
\quad \land\ state[p] = \text{“owner”} \\
\quad \land\ state' = [state \text{ EXCEPT } ![p] = \text{“idle”}] \\
\quad \land \text{ LET } msg \triangleq [TS \mapsto clock[p],\ proc \mapsto p,\ cmd \mapsto \text{“release”}] \\
\qquad \text{IN } msgQ' = [msgQ \text{ EXCEPT } ![p] = \\
\qquad\qquad\qquad [q \in Proc \setminus \{p\} \mapsto Append(@[q],\ msg)]\,] \\
\quad \land\ lastTSent' = [lastTSent \text{ EXCEPT } ![p] = [q \in Proc \setminus \{p\} \mapsto clock[p]]\,] \\
\quad \land \text{ UNCHANGED } \langle clock,\ lastTRcvd,\ reqSet \rangle
\end{array}
$$

$$
\begin{array}{l}
RcvMsg(p,\ q) \triangleq \\
\quad \text{LET } msg \triangleq Head(msgQ[q][p]) \\
\qquad msgQTail \triangleq [msgQ \text{ EXCEPT } ![q][p] = Tail(@)] \\
\qquad ack \triangleq [TS \mapsto clock'[p],\ proc \mapsto p,\ cmd \mapsto \text{“ack”}] \\
\quad \text{IN } \land\ msgQ[q][p] \neq \langle\,\rangle \\
\qquad \land\ clock' = [clock \text{ EXCEPT } ![p] = \text{IF } msg.TS > @ \text{ THEN } msg.TS \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } @\,] \\
\qquad \land \text{ IF } \land\ msg.cmd = \text{“acquire”} \\
\qquad\qquad\quad \land\ [TS \mapsto lastTSent[p][q] + 1,\ proc \mapsto p] \prec msg \\
\qquad\qquad \text{THEN } \land\ msgQ' = [msgQTail \text{ EXCEPT } ![p][q] = Append(@,\ ack)] \\
\qquad\qquad\qquad\quad \land\ lastTSent' = [lastTSent \text{ EXCEPT } ![p][q] = clock'[p]\,] \\
\qquad\qquad \text{ELSE } \land\ msgQ' = msgQTail \\
\qquad\qquad\qquad\quad \land \text{ UNCHANGED } lastTSent \\
\qquad \land\ lastTRcvd' = [lastTRcvd \text{ EXCEPT } ![p][q] = msg.TS] \\
\qquad \land\ reqSet' = [reqSet \text{ EXCEPT } ![p] = \\
\qquad\qquad\qquad \text{CASE } msg.cmd = \text{“acquire”} \to @ \cup \{msg\} \\
\qquad\qquad\qquad\quad \square\ msg.cmd = \text{“release”} \to \{m \in @ : m.proc \neq q\} \\
\qquad\qquad\qquad\quad \square\ msg.cmd = \text{“ack”} \to @\,] \\
\qquad \land \text{ UNCHANGED } state
\end{array}
$$

$$
\begin{array}{l}
Tick(p) \triangleq \land \exists\, n \in Nat : \land\ n > clock[p] \\
\qquad\qquad\qquad\qquad\qquad\quad \land\ clock' = [clock \text{ EXCEPT } ![p] = n] \\
\qquad\qquad \land \text{ UNCHANGED } \langle state,\ msgQ,\ reqSet,\ lastTSent,\ lastTRcvd \rangle
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq \exists\, p \in Proc : \lor\ Request(p) \lor Acquire(p) \lor Release(p) \\
\qquad\qquad\qquad\quad \lor \exists\, q \in Proc \setminus \{p\} : RcvMsg(p,\ q) \\
\qquad\qquad\qquad\quad \lor\ Tick(p)
\end{array}
$$

$$
\begin{array}{l}
Liveness \triangleq \forall\, p \in Proc : \land\ \mathrm{WF}_{vars}(Acquire(p)) \\
\qquad\qquad\qquad\qquad\quad \land\ \forall\, q \in Proc \setminus \{p\} : \mathrm{WF}_{vars}(RcvMsg(p,\ q))
\end{array}
$$

$$
Spec \triangleq Init \land \Box[Next]_{vars} \land Liveness
$$

MODULE *SMParameters*

CONSTANTS

$Proc$,	The set of processes.
Cmd,	The set of commands.
$InitState$,	The initial state-machine state.
$NewState(\_, \_)$,	$NewState(s, c)$ is the state obtained by executing command c in state s.
$Output(\_, \_)$	$Output(s, c)$ is the output generated by executing command c in state s.

VARIABLE *int* Represents the abstract interface.

CONSTANTS

$Request(\_, \_, \_, \_)$,	A $Request(int, int', c, p)$ step represents process p issuing command c to the system.
$Reply(\_, \_, \_, \_)$	A $Reply(int, int', r, p)$ step represents the system issuing reply r to process p.

MODULE *StateMachine*

EXTENDS *SMParams*

$Inner(state, pending, rcvd) \triangleq$ INSTANCE *InnerSM*

$Spec \triangleq \exists\, state, pending, rcvd : Inner(state, pending, rcvd)!ISpec$

MODULE *InnerSM*

EXTENDS *SMParams*, *Sequences*

VARIABLES *state*, *pending*, *rcvd*

$$vars \triangleq \langle int, state, pending, rcvd \rangle$$

$$\begin{aligned} Init \triangleq\ & \land\ state = [p \in Proc \mapsto InitState] \\ & \land\ pending = [p \in Proc \mapsto \langle\,\rangle] \\ & \land\ rcvd = [p \in Proc \mapsto \langle\,\rangle] \end{aligned}$$

$$\begin{aligned} & IssueCmd(p, c) \triangleq \\ & \quad \land\ Request(int, int', c, p) \\ & \quad \land\ rcvd' = [rcvd \text{ EXCEPT } ![p] = Append(@, c)] \\ & \quad \land\ \text{UNCHANGED } \langle state, pending \rangle \end{aligned}$$

$$\begin{aligned} & SequenceCmd(p) \triangleq \\ & \quad \land\ rcvd[p] \neq \langle\,\rangle \\ & \quad \land\ rcvd' = [rcvd \text{ EXCEPT } ![p] = Tail(@)] \\ & \quad \land\ pending' = [q \in Proc \mapsto Append(pending[q], Head(rcvd[p]))] \\ & \quad \land\ \text{UNCHANGED } \langle int, state \rangle \end{aligned}$$

$$\begin{aligned} & IssueReply(p) \triangleq \\ & \quad \land\ pending[p] \neq \langle\,\rangle \\ & \quad \land\ Reply(int, int', Output(state[p], Head(pending[p])), p) \\ & \quad \land\ state' = [state \text{ EXCEPT } ![p] = NewState(@, Head(pending[p]))] \\ & \quad \land\ pending' = [pending \text{ EXCEPT } ![p] = Tail(@)] \\ & \quad \land\ \text{UNCHANGED } rcvd \end{aligned}$$

$$\begin{aligned} Next \triangleq\ & \exists\, p \in Proc : \lor \exists\, c \in Cmd : IssueCmd(p, c) \\ & \qquad\qquad\ \lor SequenceCmd(p) \\ & \qquad\qquad\ \lor IssueReply(p) \end{aligned}$$

$$\begin{aligned} ISpec \triangleq\ & \land\ Init \\ & \land\ \Box[Next]_{vars} \\ & \land\ \forall\, p \in Proc : \land\ WF_{vars}(SequenceCmd(p)) \\ & \qquad\qquad\ \land\ WF_{vars}(IssueReply(p)) \end{aligned}$$

MODULE *MutexSM*

VARIABLE *int*

CONSTANTS *Proc*, *Request*(_, _, _, _), *Reply*(_, _, _, _)

InitState $\triangleq$ $[free \mapsto \text{TRUE}, waiting \mapsto \langle\,\rangle]$

NewState(s, c) $\triangleq$

```
IF c.cmd = "acquire"
  THEN IF s.free THEN [s EXCEPT !.free = FALSE]
                 ELSE [s EXCEPT !.waiting = Append(@, c.proc)]
  ELSE IF s.waiting = ⟨⟩ THEN [s EXCEPT !.free = TRUE]
                         ELSE [s EXCEPT !.waiting = Tail(@)]
```

None $\triangleq$ CHOOSE $v : v \notin Proc$

Output(s, c) $\triangleq$

```
IF c.cmd = "acquire" THEN IF s.free THEN c.proc
                                    ELSE None
                     ELSE IF s.waiting ≠ ⟨⟩ THEN Head(s.waiting)
                                            ELSE None
```

Cmd $\triangleq$ $[cmd : \{\text{“acquire”}, \text{“release”}\}, proc : Proc]$

INSTANCE *StateMachine*

MODULE *MutexUser*

VARIABLE *int*

CONSTANTS $Request(\_, \_, \_, \_)$, $Reply(\_, \_, \_, \_)$, *Proc*

MODULE *Inner*

VARIABLE *state*

$None \triangleq$ CHOOSE $v : v \notin Proc$

$Init \triangleq state = [p \in Proc \mapsto \text{“idle”}]$

$IssueAcq(p) \triangleq \land state[p] = \text{“idle”}$
$\land Request(int, int', [cmd \mapsto \text{“acquire”}, proc \mapsto p], p)$
$\land state' = [state \text{ EXCEPT } ![p] = \text{“waiting”}]$

$IssueRel(p) \triangleq \land state[p] = \text{“owner”}$
$\land Request(int, int' [cmd \mapsto \text{“release”}, proc \mapsto p], p)$
$\land state' = [state \text{ EXCEPT } ![p] = \text{“idle”}]$

$RcvReply(p) \triangleq \exists r \in Proc \cup \{None\} :$
$\land Reply(int, int', r, p)$
$\land state' = \text{IF } r = p$
THEN $[state \text{ EXCEPT } ![p] = \text{“owner”}]$
ELSE $state$

$Next \triangleq \exists p \in Proc : IssueAcq(p) \lor IssueRel(p) \lor RcvReply(p)$

$ISpec \triangleq Init \land \Box[Next]_{\langle int, state \rangle}$

$I(state) \triangleq$ INSTANCE *Inner*

$Spec \triangleq \exists state : I(state)!ISpec$

MODULE *MutexSystem*

VARIABLE *int*

CONSTANTS *Request*(_, _, _, _), *Reply*(_, _, _, _), *Proc*

$Env \triangleq$ INSTANCE *MutexUser*

$Sys \triangleq$ INSTANCE *MutexSM*

$ClosedSystem \triangleq Env!Spec \wedge Sys!Spec$

$OpenSystem \triangleq Env!Spec \stackrel{+}{\triangleright} Sys!Spec$

TLA home page

`http://www.research.digital.com/SRC/tla/`

Handout

`http://www.research.digital.com/SRC/tla/podc00-handout.ps`