MODULE *LamportMutex_proofs*

Proof of type correctness and safety of *Lamport*'s distributed mutual-exclusion algorithm.

EXTENDS *LamportMutex*, *SequenceTheorems*, *TLAPS*

USE DEF *Clock*

Proof of type correctness.

LEMMA $BroadcastType \triangleq$
  ASSUME $network \in [Proc \to [Proc \to Seq(Message)]]$,
    NEW $s \in Proc$, NEW $m \in Message$
  PROVE $Broadcast(s, m) \in [Proc \to Seq(Message)]$
BY *AppendProperties* DEF *Broadcast*

LEMMA $TypeCorrect \triangleq Spec \Rightarrow \Box TypeOK$
$\langle 1 \rangle 1.$ $Init \Rightarrow TypeOK$
  BY DEF *Init*, *TypeOK*
$\langle 1 \rangle 2.$ $TypeOK \land [Next]_{vars} \Rightarrow TypeOK'$
  $\langle 2 \rangle$ SUFFICES ASSUME $TypeOK$,
      $[Next]_{vars}$
    PROVE $TypeOK'$
    OBVIOUS
  $\langle 2 \rangle$.USE DEF *TypeOK*
  $\langle 2 \rangle 1.$ ASSUME NEW $p \in Proc$,
      $Request(p)$
    PROVE $TypeOK'$
    BY $\langle 2 \rangle 1$, *BroadcastType*, *Zenon* DEF *Request*, *Message*
  $\langle 2 \rangle 2.$ ASSUME NEW $p \in Proc$,
      $Enter(p)$
    PROVE $TypeOK'$
    BY $\langle 2 \rangle 2$ DEF *Enter*
  $\langle 2 \rangle 3.$ ASSUME NEW $p \in Proc$,
      $Exit(p)$
    PROVE $TypeOK'$
    BY $\langle 2 \rangle 3$, *BroadcastType*, *Zenon* DEF *Exit*, *Message*
  $\langle 2 \rangle 4.$ ASSUME NEW $p \in Proc$,
      NEW $q \in Proc \setminus \{p\}$,
      $ReceiveRequest(p, q)$
    PROVE $TypeOK'$
    $\langle 3 \rangle$.DEFINE $m \triangleq Head(network[q][p])$
      $c \triangleq m.clock$
    $\langle 3 \rangle 1.$ $\land\ network[q][p] \neq \langle\rangle$
      $\land\ m.type =$ "req"
      $\land\ req' = [req$ EXCEPT $![p][q] = c]$
      $\land\ clock' = [clock$ EXCEPT $![p] =$ IF $c > clock[p]$ THEN $c + 1$ ELSE $@ + 1]$
      $\land\ network' = [network$ EXCEPT $![q][p] = Tail(@)$,

$![p][q] = Append(@, AckMessage)]$

$\quad\land$ UNCHANGED $\langle ack, crit\rangle$

BY $\langle 2\rangle 4$ DEF *ReceiveRequest*

$\langle 3\rangle 2.$ $m \in Message$

BY $\langle 3\rangle 1$

$\langle 3\rangle 3.$ $m \in \{ReqMessage(cc) : cc \in Clock\}$

BY $\langle 3\rangle 1$, $\langle 3\rangle 2$ DEF *Message*, *AckMessage*, *RelMessage*

$\langle 3\rangle 4.$ $\land\ clock' \in [Proc \to Clock]$

$\quad\land\ req' \in [Proc \to [Proc \to Nat]]$

BY $\langle 3\rangle 1$, $\langle 3\rangle 3$ DEF *ReqMessage*

$\langle 3\rangle 5.$ $network' \in [Proc \to [Proc \to Seq(Message)]]$

$\langle 4\rangle$.DEFINE $nw \triangleq [network$ EXCEPT $![q][p] = Tail(@)]$

$\langle 4\rangle 1.$ $nw \in [Proc \to [Proc \to Seq(Message)]]$

BY $\langle 3\rangle 1$

$\langle 4\rangle$.HIDE DEF *nw*

$\langle 4\rangle 2.$ $AckMessage \in Message$

BY DEF *Message*

$\langle 4\rangle 3.$ $[nw$ EXCEPT $![p][q] = Append(@, AckMessage)] \in [Proc \to [Proc \to Seq(Message)]]$

BY $\langle 4\rangle 1$, $\langle 4\rangle 2$

$\langle 4\rangle$.QED BY $\langle 3\rangle 1$, $\langle 4\rangle 3$, *Zenon* DEF *nw*

$\langle 3\rangle 6.$ $\land\ ack' \in [Proc \to$ SUBSET $Proc]$

$\quad\land\ crit' \in$ SUBSET $Proc$

BY $\langle 3\rangle 1$

$\langle 3\rangle$.QED BY $\langle 3\rangle 4$, $\langle 3\rangle 5$, $\langle 3\rangle 6$, *Zenon*

$\langle 2\rangle 5.$ ASSUME NEW $p \in Proc$,

NEW $q \in Proc \setminus \{p\}$,

$ReceiveAck(p, q)$

PROVE $TypeOK'$

BY $\langle 2\rangle 5$ DEF *ReceiveAck*

$\langle 2\rangle 6.$ ASSUME NEW $p \in Proc$,

NEW $q \in Proc \setminus \{p\}$,

$ReceiveRelease(p, q)$

PROVE $TypeOK'$

BY $\langle 2\rangle 6$ DEF *ReceiveRelease*

$\langle 2\rangle 7.$CASE UNCHANGED *vars*

BY $\langle 2\rangle 7$ DEF *vars*

$\langle 2\rangle 8.$ QED BY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$, $\langle 2\rangle 5$, $\langle 2\rangle 6$, $\langle 2\rangle 7$ DEF *Next*

$\langle 1\rangle$.QED BY $\langle 1\rangle 1$, $\langle 1\rangle 2$, *PTL* DEF *Spec*

Inductive invariants for the algorithm.

We start the proof of safety by defining some auxiliary predicates:
- $Contains(s, mt)$ holds if channel s contains a message of type mt.
- $AtMostOne(s, mt)$ holds if channel s holds zero or one messages of type $mtype$.

- *Precedes*$(s, mt1, mt2)$ holds if in channel s, any message of type $mt1$ precedes any message of type $mt2$.

$Contains(s, mtype) \triangleq \exists\, i \in 1 \,..\, Len(s) : s[i].type = mtype$
$AtMostOne(s, mtype) \triangleq \forall\, i, j \in 1 \,..\, Len(s) :$
  $s[i].type = mtype \land s[j].type = mtype \Rightarrow i = j$
$Precedes(s, mt1, mt2) \triangleq \forall\, i, j \in 1 \,..\, Len(s) :$
  $s[i].type = mt1 \land s[j].type = mt2 \Rightarrow i < j$

LEMMA $NotContainsAtMostOne \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$, $\neg Contains(s, mtype)$
  PROVE $AtMostOne(s, mtype)$
BY DEF $Contains$, $AtMostOne$

LEMMA $NotContainsPrecedes \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mt1$, NEW $mt2$, $\neg Contains(s, mt2)$
  PROVE $\land\, Precedes(s, mt1, mt2)$
    $\land\, Precedes(s, mt2, mt1)$
BY DEF $Contains$, $Precedes$

LEMMA $PrecedesHead \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mt1$, NEW $mt2$,
    $s \neq \langle\rangle$,
    $Precedes(s, mt1, mt2)$, $Head(s).type = mt2$
  PROVE $\neg Contains(s, mt1)$
BY DEF $Precedes$, $Contains$

LEMMA $AtMostOneTail \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$,
    $s \neq \langle\rangle$, $AtMostOne(s, mtype)$
  PROVE $AtMostOne(Tail(s), mtype)$
BY DEF $AtMostOne$

LEMMA $ContainsTail \triangleq$
  ASSUME NEW $s \in Seq(Message)$, $s \neq \langle\rangle$,
    NEW $mtype$, $AtMostOne(s, mtype)$
  PROVE $Contains(Tail(s), mtype) \equiv Contains(s, mtype) \land Head(s).type \neq mtype$
BY DEF $Contains$, $AtMostOne$

LEMMA $AtMostOneHead \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$,
    $AtMostOne(s, mtype)$, $s \neq \langle\rangle$, $Head(s).type = mtype$
  PROVE $\neg Contains(Tail(s), mtype)$
$\langle 1 \rangle$.SUFFICES ASSUME NEW $i \in 1 \,..\, Len(Tail(s))$, $Tail(s)[i].type = mtype$
    PROVE FALSE
  BY $Tail(s) \in Seq(Message)$, Isa DEF $Contains$
$\langle 1 \rangle$.QED BY $HeadTailProperties$ DEF $AtMostOne$

LEMMA $ContainsSend \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$, NEW $m \in Message$
  PROVE $Contains(Append(s, m), mtype) \equiv m.type = mtype \lor Contains(s, mtype)$
BY DEF $Contains$

LEMMA $NotContainsSend \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$, $\neg Contains(s, mtype)$, NEW $m \in Message$
  PROVE $\land AtMostOne(Append(s, m), mtype)$
  $\land m.type \neq mtype \Rightarrow \neg Contains(Append(s, m), mtype)$
BY DEF $Contains$, $AtMostOne$

LEMMA $AtMostOneSend \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mtype$, $AtMostOne(s, mtype)$,
  NEW $m \in Message$, $m.type \neq mtype$
  PROVE $AtMostOne(Append(s, m), mtype)$
BY DEF $AtMostOne$

LEMMA $PrecedesSend \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mt1$, NEW $mt2$,
  NEW $m \in Message$, $m.type \neq mt1$
  PROVE $Precedes(Append(s, m), mt1, mt2) \equiv Precedes(s, mt1, mt2)$
BY DEF $Precedes$

LEMMA $PrecedesTail \triangleq$
  ASSUME NEW $s \in Seq(Message)$, NEW $mt1$, NEW $mt2$, $Precedes(s, mt1, mt2)$
  PROVE $Precedes(Tail(s), mt1, mt2)$
BY DEF $Precedes$

LEMMA $PrecedesInTail \triangleq$
  ASSUME NEW $s \in Seq(Message)$, $s \neq \langle\rangle$,
  NEW $mt1$, NEW $mt2$, $mt1 \neq mt2$,
  $Head(s).type = mt1 \lor Head(s).type \notin \{mt1, mt2\}$,
  $Precedes(Tail(s), mt1, mt2)$
  PROVE $Precedes(s, mt1, mt2)$
BY $SMTT(30)$ DEF $Precedes$

In order to prove the safety property of the algorithm, we prove two inductive invariants. Our first invariant is itself a conjunction of two predicates: - The first one states that each channel holds at most one message of
each type. Moreover, no process ever sends a message to itself.
- The second predicate describes how request, acknowledgement, and release messages are exchanged among processes, but does not refer to clock values held in the clock and *req* variables.

$NetworkInv(p, q) \triangleq$
  LET $s \triangleq network[p][q]$
  IN $\land AtMostOne(s, \text{“req”})$

$\quad\quad \land\ AtMostOne(s,\ \text{“ack”})$
$\quad\quad \land\ AtMostOne(s,\ \text{“rel”})$
$\quad\quad \land\ network[p][p] = \langle\rangle$

$CommInv(p) \ \triangleq$
$\quad \lor\ \land\ req[p][p] = 0 \land ack[p] = \{\} \land p \notin crit$
$\quad\quad \land\ \forall\, q \in Proc : \neg Contains(network[p][q],\ \text{“req”}) \land \neg Contains(network[q][p],\ \text{“ack”})$
$\quad \lor\ \land\ req[p][p] > 0 \land p \in ack[p]$
$\quad\quad \land\ p \in crit \Rightarrow ack[p] = Proc$
$\quad\quad \land\ \forall\, q \in Proc :$
$\quad\quad\quad \text{LET } pq \ \triangleq\ network[p][q]$
$\quad\quad\quad\quad\quad qp \ \triangleq\ network[q][p]$
$\quad\quad\quad \text{IN} \quad \lor\ \land\ q \in ack[p]$
$\quad\quad\quad\quad\quad\quad \land\ \neg Contains(pq,\ \text{“req”}) \land \neg Contains(qp,\ \text{“ack”}) \land \neg Contains(pq,\ \text{“rel”})$
$\quad\quad\quad\quad\quad \lor\ \land\ q \notin ack[p] \land Contains(qp,\ \text{“ack”})$
$\quad\quad\quad\quad\quad\quad \land\ \neg Contains(pq,\ \text{“req”}) \land \neg Contains(pq,\ \text{“rel”})$
$\quad\quad\quad\quad\quad \lor\ \land\ q \notin ack[p] \land Contains(pq,\ \text{“req”})$
$\quad\quad\quad\quad\quad\quad \land\ \neg Contains(qp,\ \text{“ack”}) \land Precedes(pq,\ \text{“rel”},\ \text{“req”})$

$BasicInv \ \triangleq$
$\quad \land\ \forall\, p,\ q \in Proc : NetworkInv(p,\ q)$
$\quad \land\ \forall\, p \in Proc : CommInv(p)$

$\text{THEOREM } BasicInvariant \ \triangleq\ Spec \Rightarrow \Box BasicInv$
$\langle 1 \rangle 1.\ Init \Rightarrow BasicInv$
$\quad \text{BY DEF } Init,\ BasicInv,\ CommInv,\ NetworkInv,\ Contains,\ AtMostOne$
$\langle 1 \rangle 2.\ TypeOK \land BasicInv \land [Next]_{vars} \Rightarrow BasicInv'$
$\quad \langle 2 \rangle \text{ SUFFICES ASSUME } TypeOK,\ BasicInv,\ [Next]_{vars}$
$\quad\quad\quad\quad\quad\quad \text{PROVE } BasicInv'$
$\quad\quad \text{OBVIOUS}$
$\quad \langle 2 \rangle.\text{USE DEF } TypeOK$
$\quad \langle 2 \rangle 1.\ \text{ASSUME NEW } n \in Proc,\ Request(n)$
$\quad\quad\quad \text{PROVE } BasicInv'$
$\quad\quad \langle 3 \rangle 1.\ \land\ req[n][n] = 0$
$\quad\quad\quad\quad \land\ req' = [req \text{ EXCEPT } ![n][n] = clock[n]]$
$\quad\quad\quad\quad \land\ network' = [network \text{ EXCEPT } ![n] = Broadcast(n,\ ReqMessage(clock[n]))]$
$\quad\quad\quad\quad \land\ ack' = \ [ack \text{ EXCEPT } ![n] = \{n\}]$
$\quad\quad\quad\quad \land\ crit' = crit$
$\quad\quad\quad \text{BY } \langle 2 \rangle 1 \text{ DEF } Request$
$\quad\quad \langle 3 \rangle.\ \land\ ReqMessage(clock[n]) \in Message$
$\quad\quad\quad\quad \land\ ReqMessage(clock[n]).type = \text{“req”}$
$\quad\quad\quad \text{BY DEF } ReqMessage,\ Message$
$\quad\quad \langle 3 \rangle\text{a}.\ \neg(req[n][n] > 0)$
$\quad\quad\quad \text{BY } \langle 3 \rangle 1$
$\quad\quad \langle 3 \rangle 2.\ \land\ n \notin crit$
$\quad\quad\quad\quad \land\ \forall\, q \in Proc : \neg Contains(network[n][q],\ \text{“req”}) \land \neg Contains(network[q][n],\ \text{“ack”})$

BY $\langle 3\rangle$a DEF $BasicInv$, $CommInv$
$\langle 3\rangle 3.$ ASSUME NEW $p \in Proc$, NEW $q \in Proc$
PROVE $NetworkInv(p, q)'$
BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$, $NotContainsSend$, $AtMostOneSend$ DEF $Broadcast$, $BasicInv$, $NetworkInv$
$\langle 3\rangle 4.$ ASSUME NEW $p \in Proc$
PROVE $CommInv(p)'$
$\langle 4\rangle 1.$CASE $p = n$
$\langle 5\rangle.\ \wedge req'[p][p] > 0 \wedge p \in ack'[p]$
$\wedge\ p \notin crit'$
BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 4\rangle 1$
$\langle 5\rangle.\ \wedge \neg Contains(network'[p][n], \text{“req”})$
$\wedge \neg Contains(network'[n][p], \text{“ack”})$
$\wedge \neg Contains(network'[p][n], \text{“rel”})$
BY $\langle 3\rangle 3$, $\langle 4\rangle 1$ DEF $NetworkInv$, $Contains$
$\langle 5\rangle.$ASSUME NEW $q \in Proc \setminus \{n\}$
PROVE $\wedge\ q \notin ack'[p]$
$\wedge\ Contains(network'[p][q], \text{“req”})$
$\wedge \neg Contains(network'[q][p], \text{“ack”})$
BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 4\rangle 1$, $ContainsSend$ DEF $Broadcast$
$\langle 5\rangle.\forall q \in Proc \setminus \{n\} : Precedes(network[p][q], \text{“rel”}, \text{“req”})$
BY $\langle 3\rangle 2$, $\langle 4\rangle 1$, $NotContainsPrecedes$
$\langle 5\rangle.\forall q \in Proc \setminus \{n\} : Precedes(network'[p][q], \text{“rel”}, \text{“req”})$
BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $PrecedesSend$ DEF $Broadcast$
$\langle 5\rangle.$QED BY DEF $CommInv$
$\langle 4\rangle 2.$CASE $p \neq n$
$\langle 5\rangle.CommInv(p)$
BY DEF $BasicInv$
$\langle 5\rangle.$UNCHANGED $\langle req[p][p], ack[p], crit\rangle$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$
$\langle 5\rangle.\forall q \in Proc :$ UNCHANGED $network[p][q]$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$
$\langle 5\rangle.\ \wedge \forall q \in Proc \setminus \{n\} :$ UNCHANGED $network[q][p]$
$\wedge\ p = n \Rightarrow$ UNCHANGED $network[n][p]$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$ DEF $Broadcast$
$\langle 5\rangle.n \neq p \Rightarrow Contains(network'[n][p], \text{“ack”}) \equiv Contains(network[n][p], \text{“ack”})$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$, $ContainsSend$ DEF $Broadcast$
$\langle 5\rangle.$QED BY DEF $CommInv$
$\langle 4\rangle.$QED BY $\langle 4\rangle 1$, $\langle 4\rangle 2$
$\langle 3\rangle.$QED BY $\langle 3\rangle 3$, $\langle 3\rangle 4$ DEF $BasicInv$
$\langle 2\rangle 2.$ ASSUME NEW $n \in Proc$, $Enter(n)$
PROVE $BasicInv'$
BY $\langle 2\rangle 2$ DEF $Enter$, $BasicInv$, $NetworkInv$, $CommInv$
$\langle 2\rangle 3.$ ASSUME NEW $n \in Proc$, $Exit(n)$
PROVE $BasicInv'$
$\langle 3\rangle 1.\ \wedge req[n][n] > 0$

$\land\ ack[n] = Proc$
$\land\ \forall\, q \in Proc : \ \land \neg Contains(network[n][q], \text{“req”})$
$\qquad\qquad\qquad\quad \land \neg Contains(network[q][n], \text{“ack”})$
$\qquad\qquad\qquad\quad \land \neg Contains(network[n][q], \text{“rel”})$
$\land\ network' = [network \text{ EXCEPT } ![n] =$
$\qquad [q \in Proc \mapsto \text{IF } n = q \text{ THEN } network[n][q] \text{ ELSE } Append(network[n][q], RelMessage)]$
$\land\ crit' = crit \setminus \{n\}$
$\land\ req' = [req \text{ EXCEPT } ![n][n] = 0]$
$\land\ ack' = [ack \text{ EXCEPT } ![n] = \{\}]$
$\land\ clock' = clock$

BY $\langle 2\rangle 3$ DEF $Exit$, $Broadcast$, $BasicInv$, $CommInv$

$\langle 3\rangle.\ \land\ RelMessage \in Message$
$\quad \land\ RelMessage.type = \text{“rel”}$

BY DEF $RelMessage$, $Message$

$\langle 3\rangle 2.$ ASSUME NEW $p \in Proc$, NEW $q \in Proc$
PROVE $NetworkInv(p, q)'$

$\langle 4\rangle 1.$CASE $p = n$

$\langle 5\rangle.\ \land\ AtMostOne(network'[p][q], \text{“req”})$
$\quad \land\ AtMostOne(network'[p][q], \text{“rel”})$

BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $NotContainsAtMostOne$, $NotContainsSend$

$\langle 5\rangle.AtMostOne(network[p][q], \text{“ack”})$

BY DEF $BasicInv$, $NetworkInv$

$\langle 5\rangle.AtMostOne(network'[p][q], \text{“ack”})$

BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $AtMostOneSend$

$\langle 5\rangle.network'[p][p] = \langle\rangle$

BY $\langle 3\rangle 1$, $\langle 4\rangle 1$ DEF $BasicInv$, $NetworkInv$

$\langle 5\rangle.$QED BY DEF $NetworkInv$

$\langle 4\rangle 2.$CASE $p \neq n$

$\langle 5\rangle.\ \land\ network'[p][p] = network[p][p]$
$\quad \land\ network'[p][q] = network[p][q]$

BY $\langle 3\rangle 1$, $\langle 4\rangle 2$

$\langle 5\rangle.$QED BY DEF $BasicInv$, $NetworkInv$

$\langle 4\rangle.$QED BY $\langle 4\rangle 1$, $\langle 4\rangle 2$

$\langle 3\rangle 3.$ ASSUME NEW $p \in Proc$
PROVE $CommInv(p)'$

$\langle 4\rangle 1.$CASE $p = n$

BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $NotContainsSend$ DEF $CommInv$

$\langle 4\rangle 2.$CASE $p \neq n$

$\langle 5\rangle.\ \land\ req'[p][p] = req[p][p]$
$\quad \land\ ack'[p] = ack[p]$
$\quad \land\ (p \in crit') \equiv (p \in crit)$
$\quad \land\ \forall\, q \in Proc : network'[p][q] = network[p][q]$

BY $\langle 3\rangle 1$, $\langle 4\rangle 2$

$\langle 5\rangle.$ASSUME NEW $q \in Proc$
PROVE $Contains(network'[q][p], \text{“ack”}) \equiv Contains(network[q][p], \text{“ack”})$

$\langle 6\rangle 1.$ CASE $n = q$
  $\langle 7\rangle.$ $network'[q][p] = Append(network[q][p],\ RelMessage)$
    BY $\langle 3\rangle 1,\ \langle 4\rangle 2,\ \langle 6\rangle 1$
  $\langle 7\rangle.$QED BY $ContainsSend$
$\langle 6\rangle 2.$ CASE $n \neq q$
  BY $\langle 3\rangle 1,\ \langle 6\rangle 2$
$\langle 6\rangle.$QED BY $\langle 6\rangle 1,\ \langle 6\rangle 2$
$\langle 5\rangle.$QED BY DEF $BasicInv,\ CommInv$
$\langle 4\rangle.$QED BY $\langle 4\rangle 1,\ \langle 4\rangle 2$
$\langle 3\rangle.$QED BY $\langle 3\rangle 2,\ \langle 3\rangle 3$ DEF $BasicInv$

$\langle 2\rangle 4.$ ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveRequest(n,\ k)$
      PROVE $BasicInv'$

$\langle 3\rangle 1.$ $\land\ network[k][n] \neq \langle\rangle$
  $\land$ LET $m \triangleq Head(network[k][n])$
    IN $\land\ m.type = \text{“req”}$
      $\land\ \forall\, p \in Proc : req'[p][p] = req[p][p]$
      $\land\ network' = [network$ EXCEPT $![k][n] = Tail(network[k][n]),$
        $![n][k] = Append(network[n][k],\ AckMessage)]$
      $\land$ UNCHANGED $\langle ack,\ crit\rangle$
  BY $\langle 2\rangle 4$ DEF $ReceiveRequest$
$\langle 3\rangle 2.$ $Contains(network[k][n],\ \text{“req”})$
  BY $\langle 3\rangle 1$ DEF $Contains$
$\langle 3\rangle 3.$ $\land\ req[k][k] > 0 \land k \in ack[k]$
  $\land\ k \in crit \Rightarrow ack[k] = Proc$
  $\land\ n \notin ack[k]$
  $\land\ \neg Contains(network[n][k],\ \text{“ack”}) \land \neg Contains(network[k][n],\ \text{“rel”})$
  BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ PrecedesHead$ DEF $BasicInv,\ CommInv$
$\langle 3\rangle.$ $\land\ AckMessage \in Message$
  $\land\ AckMessage.type = \text{“ack”}$
  BY DEF $AckMessage,\ Message$
$\langle 3\rangle 4.$ ASSUME NEW $p \in Proc$, NEW $q \in Proc$
    PROVE $NetworkInv(p,\ q)'$
  $\langle 4\rangle 1.$ $AtMostOne(network'[p][q],\ \text{“req”})$
    BY $\langle 3\rangle 1,\ AtMostOneTail,\ AtMostOneSend,\ Zenon$ DEF $BasicInv,\ NetworkInv$
  $\langle 4\rangle 2.$ $AtMostOne(network'[p][q],\ \text{“ack”})$
    $\langle 5\rangle.$DEFINE $nw \triangleq [network$ EXCEPT $![k][n] = Tail(network[k][n])]$
    $\langle 5\rangle 1.$ $\land\ nw \in [Proc \to [Proc \to Seq(Message)]]$
      $\land\ AtMostOne(nw[p][q],\ \text{“ack”})$
      $\land\ \neg Contains(nw[n][k],\ \text{“ack”})$
      BY $\langle 3\rangle 1,\ \langle 3\rangle 3,\ AtMostOneTail$ DEF $BasicInv,\ NetworkInv$
    $\langle 5\rangle.$HIDE DEF nw
    $\langle 5\rangle.$DEFINE $nw2 \triangleq [nw$ EXCEPT $![n][k] = Append(network[n][k],\ AckMessage)]$
    $\langle 5\rangle 5.$ $AtMostOne(nw2[p][q],\ \text{“ack”})$
      BY $\langle 3\rangle 3,\ \langle 5\rangle 1,\ NotContainsSend$
    $\langle 5\rangle.$QED BY $\langle 3\rangle 1,\ \langle 5\rangle 5$ DEF nw

$\langle 4 \rangle 3.$ $AtMostOne(network'[p][q], \text{“rel”})$
BY $\langle 3 \rangle 1$, $AtMostOneTail$, $AtMostOneSend$, $Zenon$ DEF $BasicInv$, $NetworkInv$
$\langle 4 \rangle 4.$ $network'[p][p] = \langle\rangle$
BY $\langle 3 \rangle 1$ DEF $BasicInv$, $NetworkInv$
$\langle 4 \rangle$.QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$, $\langle 4 \rangle 4$ DEF $NetworkInv$
$\langle 3 \rangle 5.$ ASSUME NEW $p \in Proc$
PROVE $CommInv(p)'$
$\langle 4 \rangle 1.$CASE $p = k$
$\langle 5 \rangle$.SUFFICES ASSUME NEW $q \in Proc$
PROVE $CommInv(p)!2!3!(q)'$
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$ DEF $CommInv$
$\langle 5 \rangle$.DEFINE $pq \triangleq network[p][q]$
$qp \triangleq network[q][p]$
$\langle 5 \rangle 1.$CASE $q = n$
$\langle 6 \rangle . q \notin ack'[p]$
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$, $\langle 5 \rangle 1$
$\langle 6 \rangle . \land pq \neq \langle\rangle \land pq' = Tail(pq)$
$\land qp' = Append(qp, AckMessage)$
$\land AtMostOne(pq, \text{“req”}) \land Head(pq).type = \text{“req”}$
$\land \neg Contains(pq, \text{“rel”})$
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$, $\langle 5 \rangle 1$ DEF $BasicInv$, $NetworkInv$
$\langle 6 \rangle . \land Contains(qp', \text{“ack”})$
$\land \neg Contains(pq', \text{“req”})$
$\land \neg Contains(pq', \text{“rel”})$
BY $ContainsSend$, $AtMostOneHead$, $ContainsTail$ DEF $BasicInv$, $NetworkInv$
$\langle 6 \rangle$.QED OBVIOUS
$\langle 5 \rangle 2.$CASE $q \neq n$
$\langle 6 \rangle . pq' = pq \land qp' = qp \land ack'[p] = ack[p]$
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$, $\langle 5 \rangle 2$
$\langle 6 \rangle . CommInv(p)!2!3!(q)$
BY $\langle 3 \rangle 3$, $\langle 4 \rangle 1$ DEF $BasicInv$, $CommInv$
$\langle 6 \rangle$.QED OBVIOUS
$\langle 5 \rangle$.QED BY $\langle 5 \rangle 1$, $\langle 5 \rangle 2$
$\langle 4 \rangle 2.$CASE $p = n$
$\langle 5 \rangle$.UNCHANGED $\langle req[p][p], ack[p], crit \rangle$ BY $\langle 3 \rangle 1$
$\langle 5 \rangle$.ASSUME NEW $q \in Proc$
PROVE $\land Contains(network'[p][q], \text{“req”}) \equiv Contains(network[p][q], \text{“req”})$
$\land Contains(network'[p][q], \text{“rel”}) \equiv Contains(network[p][q], \text{“rel”})$
$\land Contains(network'[q][p], \text{“ack”}) \equiv Contains(network[q][p], \text{“ack”})$
$\land Precedes(network'[p][q], \text{“rel”}, \text{“req”}) \equiv Precedes(network[p][q], \text{“rel”}, \text{“req”})$
$\langle 6 \rangle 1.$CASE $q = k$
$\langle 7 \rangle . \land network'[p][q] = Append(network[p][q], AckMessage)$
$\land network[q][p] \neq \langle\rangle \land Head(network[q][p]).type = \text{“req”}$
$\land network'[q][p] = Tail(network[q][p])$
BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$, $\langle 6 \rangle 1$

$\langle 7\rangle$.QED BY *ContainsSend*, *ContainsTail*, *PrecedesSend* DEF *BasicInv*, *NetworkInv*
$\langle 6\rangle$2.CASE $q \neq k$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$, $\langle 6\rangle 2$
$\langle 6\rangle$.QED BY $\langle 6\rangle 1$, $\langle 6\rangle 2$
$\langle 5\rangle$.QED BY DEF *BasicInv*, *CommInv*
$\langle 4\rangle$3.CASE $p \notin \{k, n\}$ all relevant variables are unchanged
$\langle 5\rangle$.$\forall q \in Proc$: UNCHANGED $\langle req[p][p], ack, crit, network[p][q], network[q][p]\rangle$
BY $\langle 3\rangle 1$, $\langle 4\rangle 3$
$\langle 5\rangle$.QED BY DEF *BasicInv*, *CommInv*
$\langle 4\rangle$.QED BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 3$
$\langle 3\rangle$.QED BY $\langle 3\rangle 4$, $\langle 3\rangle 5$ DEF *BasicInv*
$\langle 2\rangle$5. ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveAck(n, k)$
PROVE $BasicInv'$
$\langle 3\rangle$1. $\wedge\ network[k][n] \neq \langle\rangle$
$\wedge\ Head(network[k][n]).type = \text{"ack"}$
$\wedge\ ack' = [ack$ EXCEPT $![n] = @ \cup \{k\}]$
$\wedge\ network' = [network$ EXCEPT $![k][n] = Tail(@)]$
$\wedge$ UNCHANGED $\langle req, crit\rangle$
BY $\langle 2\rangle 5$ DEF *ReceiveAck*
$\langle 3\rangle$2. $Contains(network[k][n], \text{"ack"})$
BY $\langle 3\rangle 1$ DEF *Contains*
$\langle 3\rangle$3. $\wedge\ req[n][n] > 0 \wedge n \in ack[n]$
$\wedge\ n \in crit \Rightarrow ack[n] = Proc$
$\wedge\ k \notin ack[n]$
$\wedge\ \neg Contains(network[n][k], \text{"req"}) \wedge \neg Contains(network[n][k], \text{"rel"})$
BY $\langle 3\rangle 2$ DEF *BasicInv*, *CommInv*
$\langle 3\rangle$4. ASSUME NEW $p \in Proc$, NEW $q \in Proc$
PROVE $NetworkInv(p, q)'$
BY $\langle 3\rangle 1$, *AtMostOneTail* DEF *BasicInv*, *NetworkInv*
$\langle 3\rangle$5. ASSUME NEW $p \in Proc$
PROVE $CommInv(p)'$
$\langle 4\rangle$1.CASE $p = n$
$\langle 5\rangle$.SUFFICES ASSUME NEW $q \in Proc$, $CommInv(p)!2!3!(q)$
PROVE $CommInv(p)!2!3!(q)'$
BY $\langle 3\rangle 1$, $\langle 3\rangle 3$, $\langle 4\rangle 1$, $\neg(req[n][n] = 0)$ DEF *BasicInv*, *CommInv*
$\langle 5\rangle$1.CASE $q = k$
$\langle 6\rangle$. $\wedge\ q \in ack'[p]$
$\wedge\ \neg Contains(network'[p][q], \text{"req"})$
$\wedge\ \neg Contains(network'[p][q], \text{"rel"})$
BY $\langle 3\rangle 1$, $\langle 3\rangle 3$, $\langle 4\rangle 1$, $\langle 5\rangle 1$ DEF *BasicInv*, *NetworkInv*
$\langle 6\rangle$.$\neg Contains(network'[q][p], \text{"ack"})$
BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $\langle 5\rangle 1$, *AtMostOneHead*, *Zenon* DEF *BasicInv*, *NetworkInv*
$\langle 6\rangle$.QED OBVIOUS
$\langle 5\rangle$2.CASE $q \neq k$
BY $\langle 3\rangle 1$, $\langle 3\rangle 3$, $\langle 4\rangle 1$, $\langle 5\rangle 2$

$\langle 5 \rangle$.QED BY $\langle 5 \rangle 1$, $\langle 5 \rangle 2$

$\langle 4 \rangle 2$.CASE $p = k$

$\langle 5 \rangle$.ASSUME NEW $q \in Proc$

PROVE $\land\ Contains(network'[p][q], \text{"req"}) \equiv Contains(network[p][q], \text{"req"})$

$\land\ Contains(network'[p][q], \text{"rel"}) \equiv Contains(network[p][q], \text{"rel"})$

$\land\ Precedes(network[p][q], \text{"rel"}, \text{"req"}) \Rightarrow Precedes(network'[p][q], \text{"rel"}, \text{"req"})$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$, $ContainsTail$, $PrecedesTail$ DEF $BasicInv$, $NetworkInv$

$\langle 5 \rangle$.QED BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$ DEF $BasicInv$, $CommInv$

$\langle 4 \rangle 3$.CASE $p \notin \{n, k\}$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$ DEF $BasicInv$, $CommInv$

$\langle 4 \rangle$.QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$

$\langle 3 \rangle$.QED BY $\langle 3 \rangle 4$, $\langle 3 \rangle 5$ DEF $BasicInv$

$\langle 2 \rangle 6$. ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveRelease(n, k)$

PROVE $BasicInv'$

$\langle 3 \rangle 1$. $\land\ network[k][n] \neq \langle\rangle$

$\land\ Head(network[k][n]).type = \text{"rel"}$

$\land\ req' = [req$ EXCEPT $![n][k] = 0]$

$\land\ network' = [network$ EXCEPT $![k][n] = Tail(@)]$

$\land$ UNCHANGED $\langle ack, crit \rangle$

BY $\langle 2 \rangle 6$ DEF $ReceiveRelease$

$\langle 3 \rangle 2$. ASSUME NEW $p \in Proc$, NEW $q \in Proc$

PROVE $NetworkInv(p, q)'$

BY $\langle 3 \rangle 1$, $AtMostOneTail$ DEF $BasicInv$, $NetworkInv$

$\langle 3 \rangle 3$. ASSUME NEW $p \in Proc$, $CommInv(p)$

PROVE $CommInv(p)'$

$\langle 4 \rangle$.ASSUME NEW $q \in Proc$

PROVE $\land\ Contains(network'[p][q], \text{"req"}) \equiv Contains(network[p][q], \text{"req"})$

$\land\ Contains(network'[q][p], \text{"ack"}) \equiv Contains(network[q][p], \text{"ack"})$

$\land\ Precedes(network[p][q], \text{"rel"}, \text{"req"}) \Rightarrow Precedes(network'[p][q], \text{"rel"}, \text{"req"})$

BY $\langle 3 \rangle 1$, $ContainsTail$, $PrecedesTail$ DEF $BasicInv$, $NetworkInv$

$\langle 4 \rangle$.$Contains(network[k][n], \text{"rel"})$

BY $\langle 3 \rangle 1$ DEF $Contains$

$\langle 4 \rangle$.ASSUME NEW $q \in Proc$, $p \neq k \lor q \neq n$

PROVE $Contains(network'[p][q], \text{"rel"}) \equiv Contains(network[p][q], \text{"rel"})$

BY $\langle 3 \rangle 1$

$\langle 4 \rangle$.QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$ DEF $CommInv$

$\langle 3 \rangle$.QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ DEF $BasicInv$

$\langle 2 \rangle 7$.CASE UNCHANGED $vars$

BY $\langle 2 \rangle 7$ DEF $vars$, $BasicInv$, $CommInv$, $NetworkInv$

$\langle 2 \rangle 8$. QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$ DEF $Next$

$\langle 1 \rangle$.QED BY $TypeCorrect$, $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, PTL DEF $Spec$

The second invariant relates the clock values stored in the clock and *req* variables, as well as in request messages. Its proof relies on the "basic" invariant proved previously.

$ClockInvInner(p, q) \triangleq$

$\quad \text{LET } pq \triangleq network[p][q]$

$\qquad\quad qp \triangleq network[q][p]$

$\quad \text{IN } \land \forall i \in 1 \,..\, Len(pq) : pq[i].type = \text{“req”} \Rightarrow pq[i].clock = req[p][p]$

$\qquad\quad \land Contains(qp, \text{“ack”}) \lor q \in ack[p] \Rightarrow$

$\qquad\qquad\quad \land req[q][p] = req[p][p]$

$\qquad\qquad\quad \land clock[q] > req[p][p]$

$\qquad\qquad\quad \land Precedes(qp, \text{“ack”}, \text{“req”}) \Rightarrow$

$\qquad\qquad\qquad \forall i \in 1 \,..\, Len(qp) : qp[i].type = \text{“req”} \Rightarrow qp[i].clock > req[p][p]$

$\qquad\quad \land p \in crit \Rightarrow beats(p, q)$

$ClockInv \triangleq \forall p \in Proc : \forall q \in Proc \setminus \{p\} : ClockInvInner(p, q)$

$\text{THEOREM } ClockInvariant \triangleq Spec \Rightarrow \Box ClockInv$

$\langle 1\rangle 1.\ Init \Rightarrow ClockInv$

$\quad \text{BY DEF } Init, ClockInv, ClockInvInner, Contains$

$\langle 1\rangle 2.\ TypeOK \land BasicInv \land ClockInv \land [Next]_{vars} \Rightarrow ClockInv'$

$\quad \langle 2\rangle \text{ SUFFICES ASSUME } TypeOK, BasicInv, ClockInv, [Next]_{vars}$

$\qquad\qquad\qquad \text{PROVE } ClockInv'$

$\qquad \text{OBVIOUS}$

$\quad \langle 2\rangle.\text{USE DEF } TypeOK$

$\quad \langle 2\rangle 1.\ \text{ASSUME NEW } n \in Proc, Request(n)$

$\qquad\quad \text{PROVE } ClockInv'$

$\qquad \langle 3\rangle 1.\ \land req[n][n] = 0$

$\qquad\qquad \land req' = [req \text{ EXCEPT } ![n][n] = clock[n]]$

$\qquad\qquad \land network' = [network \text{ EXCEPT } ![n] = Broadcast(n, ReqMessage(clock[n]))]$

$\qquad\qquad \land ack' = [ack \text{ EXCEPT } ![n] = \{n\}]$

$\qquad\qquad \land \text{UNCHANGED } \langle clock, crit\rangle$

$\qquad\qquad \land n \notin crit$

$\qquad\qquad \land \forall q \in Proc : \neg Contains(network[n][q], \text{“req”}) \land \neg Contains(network[q][n], \text{“ack”})$

$\qquad\quad \text{BY } \langle 2\rangle 1 \text{ DEF } Request, BasicInv, CommInv$

$\qquad \langle 3\rangle.\ \land ReqMessage(clock[n]) \in Message$

$\qquad\qquad \land ReqMessage(clock[n]).type = \text{“req”}$

$\qquad\qquad \land ReqMessage(clock[n]).clock = req'[n][n]$

$\qquad\quad \text{BY } \langle 3\rangle 1 \text{ DEF } ReqMessage, Message$

$\qquad \langle 3\rangle 2.\ \text{ASSUME NEW } p \in Proc, \text{NEW } q \in Proc \setminus \{p\}$

$\qquad\qquad \text{PROVE } ClockInvInner(p, q)'$

$\qquad\quad \langle 4\rangle 1.\text{CASE } p = n$

$\qquad\qquad \langle 5\rangle 1.\ \text{ASSUME NEW } i \in 1 \,..\, Len(network'[p][q]), network'[p][q][i].type = \text{“req”}$

$\qquad\qquad\qquad \text{PROVE } network'[p][q][i].clock = req'[p][p]$

$\qquad\qquad\quad \text{BY } \langle 3\rangle 1, \langle 4\rangle 1, \langle 5\rangle 1 \text{ DEF } Broadcast, Contains$

$\qquad\qquad \langle 5\rangle 2.\ \neg Contains(network'[q][p], \text{“ack”}) \land q \notin ack'[p]$

$\qquad\qquad\quad \text{BY } \langle 3\rangle 1, \langle 4\rangle 1 \text{ DEF } Broadcast$

$\qquad\qquad \langle 5\rangle 3.\ p \notin crit'$

$\qquad\qquad\quad \text{BY } \langle 3\rangle 1, \langle 4\rangle 1$

$\langle 5\rangle$.QED BY $\langle 5\rangle 1$, $\langle 5\rangle 2$, $\langle 5\rangle 3$ DEF *ClockInvInner*
$\langle 4\rangle 2$.CASE $q = n$
$\langle 5\rangle 1$. $ClockInvInner(p, q)$
BY DEF *ClockInv*
$\langle 5\rangle 2$. UNCHANGED $\langle network[p][q], req[p][p], req[p][q], req[q][p], ack[p], crit\rangle$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$
$\langle 5\rangle$.DEFINE $qp \triangleq network[q][p]$
$\langle 5\rangle 3$. ASSUME $Contains(qp', \text{“ack”}) \lor q \in ack'[p]$
PROVE $\land\; req'[q][p] = req[p][p]$
$\land\; clock'[q] > req'[p][p]$
$\land\; Precedes(qp', \text{“ack”}, \text{“req”}) \Rightarrow$
$\forall\, i \in 1\,..\,Len(qp') : qp'[i].type = \text{“req”} \Rightarrow qp'[i].clock > req'[p][p]$
$\langle 6\rangle$. $Contains(qp, \text{“ack”}) \lor q \in ack[p]$
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$, $\langle 5\rangle 3$, *ContainsSend* DEF *Broadcast*
$\langle 6\rangle$.QED
BY $\langle 3\rangle 1$, $\langle 4\rangle 2$, $\langle 5\rangle 1$ DEF *ClockInvInner*, *Broadcast*, *Contains*
$\langle 5\rangle$.QED BY $\langle 5\rangle 1$, $\langle 5\rangle 2$, $\langle 5\rangle 3$ DEF *ClockInvInner*, *beats*
$\langle 4\rangle 3$.CASE $n \notin \{p, q\}$ all relevant variables unchanged
BY $\langle 3\rangle 1$, $\langle 4\rangle 3$ DEF *ClockInv*, *ClockInvInner*, *beats*
$\langle 4\rangle$.QED BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 3$
$\langle 3\rangle$.QED BY $\langle 3\rangle 2$ DEF *ClockInv*
$\langle 2\rangle 2$. ASSUME NEW $n \in Proc$, $Enter(n)$
PROVE $ClockInv'$
$\langle 3\rangle$.SUFFICES ASSUME NEW $p \in Proc$, NEW $q \in Proc \setminus \{p\}$
PROVE $ClockInvInner(p, q)'$
BY DEF *ClockInv*
$\langle 3\rangle 1$.CASE $p = n$
BY $\langle 2\rangle 2$, $\langle 3\rangle 1$ DEF *Enter*, *ClockInv*, *ClockInvInner*, *beats*
$\langle 3\rangle 2$.CASE $p \neq n$
BY $\langle 2\rangle 2$, $\langle 3\rangle 2$ DEF *Enter*, *ClockInv*, *ClockInvInner*, *beats*
$\langle 3\rangle$.QED BY $\langle 3\rangle 1$, $\langle 3\rangle 2$
$\langle 2\rangle 3$. ASSUME NEW $n \in Proc$, $Exit(n)$
PROVE $ClockInv'$
$\langle 3\rangle 1$. $\land\; n \in crit$
$\land\; crit' = crit \setminus \{n\}$
$\land\; network' = [network \text{ EXCEPT } ![n] = Broadcast(n, RelMessage)]$
$\land\; req' = [req \text{ EXCEPT } ![n][n] = 0]$
$\land\; ack' = [ack \text{ EXCEPT } ![n] = \{\}]$
$\land\; clock' = clock$
$\land\; \forall\, q \in Proc : \land\; \neg Contains(network[n][q], \text{“req”})$
$\land\; \neg Contains(network[q][n], \text{“ack”})$
BY $\langle 2\rangle 3$ DEF *Exit*, *BasicInv*, *CommInv*
$\langle 3\rangle$. $RelMessage \in Message \land RelMessage.type = \text{“rel”}$
BY DEF *RelMessage*, *Message*
$\langle 3\rangle 2$. ASSUME NEW $p \in Proc$, NEW $q \in Proc \setminus \{p\}$

PROVE $ClockInvInner(p, q)'$

$\langle 4 \rangle 1.$ CASE $n = p$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$ DEF $Broadcast$, $ClockInvInner$, $Contains$

$\langle 4 \rangle 2.$ CASE $n \neq p$

$\langle 5 \rangle 1.$ $\land$ UNCHANGED $\langle network[p][q], req[p][p], req[q][p], req[p][q], ack[p], clock \rangle$

$\land\ p \in crit' \equiv p \in crit$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$

$\langle 5 \rangle 2.$ $n \neq q \Rightarrow network'[q][p] = network[q][p]$

BY $\langle 3 \rangle 1$ DEF $Broadcast$

$\langle 5 \rangle 3.$ $\land\ Contains(network'[n][p], \text{"ack"}) \equiv Contains(network[n][p], \text{"ack"})$

$\land\ Precedes(network'[n][p], \text{"ack"}, \text{"req"}) \equiv Precedes(network[n][p], \text{"ack"}, \text{"req"})$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$, $ContainsSend$, $PrecedesSend$ DEF $Broadcast$

$\langle 5 \rangle 4.$ $\forall\, i \in 1 \,..\, Len(network'[n][p]) : network'[n][p][i].type \neq \text{"req"}$

BY $\langle 3 \rangle 1$ DEF $Broadcast$, $Contains$

$\langle 5 \rangle$.QED BY $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, $\langle 5 \rangle 3$, $\langle 5 \rangle 4$ DEF $ClockInv$, $ClockInvInner$, $beats$

$\langle 4 \rangle$.QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$

$\langle 3 \rangle$.QED BY $\langle 3 \rangle 2$ DEF $ClockInv$

$\langle 2 \rangle 4.$ ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveRequest(n, k)$

PROVE $ClockInv'$

$\langle 3 \rangle$.DEFINE $m \triangleq Head(network[k][n])$

$\langle 3 \rangle 1.$ $\land\ network[k][n] \neq \langle \rangle$

$\land\ m.type = \text{"req"}$

$\land\ req' = [req$ EXCEPT $![n][k] = m.clock]$

$\land\ clock' = [clock$ EXCEPT $![n] =$ IF $m.clock > clock[n]$ THEN $m.clock + 1$

ELSE $clock[n] + 1]$

$\land\ network' = [network$ EXCEPT $![k][n] = Tail(@)$,

$![n][k] = Append(@, AckMessage)]$

$\land$ UNCHANGED $\langle ack, crit \rangle$

$\land\ Contains(network[k][n], \text{"req"})$

BY $\langle 2 \rangle 4$ DEF $ReceiveRequest$, $ClockInv$, $ClockInvInner$, $Contains$

$\langle 3 \rangle 2.$ $m.clock = req[k][k]$

BY $\langle 3 \rangle 1$ DEF $ClockInv$, $ClockInvInner$, $Contains$

$\langle 3 \rangle 3.$ $\land\ req[k][k] > 0$

$\land\ n \notin ack[k] \land k \notin crit$

BY $\langle 3 \rangle 1$ DEF $BasicInv$, $CommInv$

$\langle 3 \rangle.$ $AckMessage \in Message \land AckMessage.type = \text{"ack"}$

BY DEF $AckMessage$, $Message$

$\langle 3 \rangle 4.$ ASSUME NEW $p \in Proc$, NEW $q \in Proc \setminus \{p\}$

PROVE $ClockInvInner(p, q)'$

$\langle 4 \rangle$.DEFINE $pq \triangleq network[p][q]$

$qp \triangleq network[q][p]$

$\langle 4 \rangle.$ $\land\ ClockInvInner(p, q)$

$\land$ UNCHANGED $req[p][p]$

BY $\langle 3 \rangle 1$ DEF $ClockInv$

$\langle 4 \rangle 1.$ CASE $p = k \land q = n$

$\langle 5 \rangle 1.$ $pq' = Tail(pq)$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$, *Zenon*
$\langle 5 \rangle 2.$ $\neg Contains(pq', \text{"req"})$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$, $\langle 5 \rangle 1$, *ContainsTail* DEF *BasicInv*, *NetworkInv*
$\langle 5 \rangle 3.$ $clock'[q] > req'[p][p]$
  BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$
$\langle 5 \rangle 4.$ $\land\ req'[q][p] = req'[p][p]$
  $\land\ q \notin ack'[p]$
  $\land\ p \notin crit'$
  BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$
$\langle 5 \rangle 5.$ ASSUME $Precedes(qp', \text{"ack"}, \text{"req"})$,
  NEW $i \in 1 \,..\, Len(qp')$, $qp'[i].type = \text{"req"}$
  PROVE FALSE
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$, $\langle 5 \rangle 5$ DEF *Precedes*
$\langle 5 \rangle$.QED BY $\langle 5 \rangle 2$, $\langle 5 \rangle 3$, $\langle 5 \rangle 4$, $\langle 5 \rangle 5$ DEF *ClockInvInner*, *Contains*
$\langle 4 \rangle 2.$CASE $p = k \land q \neq n$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 2$ DEF *ClockInvInner*, *beats*
$\langle 4 \rangle 3.$CASE $p = n \land q = k$
$\langle 5 \rangle 1.$ UNCHANGED $\langle req[q][p], clock[q], ack \rangle$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$
$\langle 5 \rangle 2.$ ASSUME NEW $i \in 1 \,..\, Len(pq')$, $pq'[i].type = \text{"req"}$
  PROVE $i \in 1 \,..\, Len(pq) \land pq'[i] = pq[i]$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$, $\langle 5 \rangle 2$
$\langle 5 \rangle 3.$ $qp' = Tail(qp) \land Head(qp).type = \text{"req"} \land qp \neq \langle\rangle$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$, *Zenon*
$\langle 5 \rangle 4.$ $Contains(qp', \text{"ack"}) \equiv Contains(qp, \text{"ack"})$
  BY $\langle 5 \rangle 3$, *ContainsTail* DEF *BasicInv*, *NetworkInv*
$\langle 5 \rangle 5.$ $\neg Contains(qp', \text{"req"})$
  BY $\langle 5 \rangle 3$, *ContainsTail* DEF *BasicInv*, *NetworkInv*
$\langle 5 \rangle 7.$ ASSUME $p \in crit'$
  PROVE $beats(p, q)'$
$\langle 6 \rangle.$ $\land\ p \in crit$
  $\land\ q \in ack[p]$
  $\land\ \neg Contains(qp, \text{"ack"})$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$, $\langle 5 \rangle 7$ DEF *BasicInv*, *CommInv*
$\langle 6 \rangle.$$Precedes(qp, \text{"ack"}, \text{"req"})$
  BY *NotContainsPrecedes*
$\langle 6 \rangle.$$req'[p][q] > req[p][p]$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 3$, $m = qp[1]$, $1 \in 1 \,..\, Len(qp)$ DEF *ClockInvInner*
$\langle 6 \rangle$.QED BY DEF *beats*
$\langle 5 \rangle$.QED BY $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, $\langle 5 \rangle 4$, $\langle 5 \rangle 5$, $\langle 5 \rangle 7$, *Zenon* DEF *ClockInvInner*, *Contains*
$\langle 4 \rangle 4.$CASE $p = n \land q \neq k$
  BY $\langle 3 \rangle 1$, $\langle 4 \rangle 4$ DEF *ClockInvInner*, *beats*
$\langle 4 \rangle 5.$CASE $p \notin \{n, k\} \land q = n$
$\langle 5 \rangle.$UNCHANGED $\langle pq, qp, req[p][q], req[q][p], ack, crit \rangle$

BY ⟨3⟩1, ⟨4⟩5

⟨5⟩.$clock[q] > req[p][p] \Rightarrow clock'[q] > req[p][p]$

BY ⟨3⟩1, ⟨3⟩2, ⟨4⟩5

⟨5⟩.QED BY DEF *ClockInvInner*, *beats*

⟨4⟩6.CASE $p \notin \{n, k\} \land q \neq n$

BY ⟨3⟩1, ⟨4⟩6, *Zenon* DEF *ClockInvInner*, *beats*

⟨4⟩.QED BY ⟨4⟩1, ⟨4⟩2, ⟨4⟩3, ⟨4⟩4, ⟨4⟩5, ⟨4⟩6

⟨3⟩.QED BY ⟨3⟩4 DEF *ClockInv*

⟨2⟩5. ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveAck(n, k)$
PROVE $ClockInv'$

⟨3⟩1. $\land\ network[k][n] \neq \langle\rangle$
$\land\ Head(network[k][n]).type = \text{“ack”}$
$\land\ ack' = [ack \text{ EXCEPT } ![n] = @ \cup \{k\}]$
$\land\ network' = [network \text{ EXCEPT } ![k][n] = Tail(@)]$
$\land$ UNCHANGED $\langle clock, req, crit\rangle$
$\land\ Contains(network[k][n], \text{“ack”})$

BY ⟨2⟩5 DEF *ReceiveAck*, *BasicInv*, *CommInv*, *Contains*

⟨3⟩2. ASSUME NEW $p \in Proc$, NEW $q \in Proc \setminus \{p\}$, $ClockInvInner(p, q)$
PROVE $ClockInvInner(p, q)'$

⟨4⟩.DEFINE $pq \triangleq network[p][q]$
$qp \triangleq network[q][p]$

⟨4⟩1.CASE $p = n \land q = k$

⟨5⟩1. $\land\ qp \neq \langle\rangle$
$\land\ Head(qp).type = \text{“ack”}$
$\land\ Contains(qp, \text{“ack”})$
$\land\ qp' = Tail(qp)$
$\land$ UNCHANGED $\langle pq, clock, req, crit\rangle$

BY ⟨3⟩1, ⟨4⟩1

⟨5⟩2. ASSUME $Precedes(qp', \text{“ack”}, \text{“req”})$
PROVE $Precedes(qp, \text{“ack”}, \text{“req”})$

BY ⟨5⟩1, ⟨5⟩2, *PrecedesInTail*, *Zenon*

⟨5⟩3. ASSUME NEW $i \in 1 \,..\, Len(qp')$, $qp'[i].type = \text{“req”}$
PROVE $i + 1 \in 1 \,..\, Len(qp) \land qp'[i] = qp[i + 1]$

BY ⟨5⟩1

⟨5⟩.QED BY ⟨3⟩2, ⟨5⟩1, ⟨5⟩2, ⟨5⟩3 DEF *ClockInvInner*, *beats*

⟨4⟩2.CASE $p = k \land q = n$

⟨5⟩1. UNCHANGED $\langle qp, ack[p], clock, req, crit\rangle$

BY ⟨3⟩1, ⟨4⟩2

⟨5⟩2. ASSUME NEW $i \in 1 \,..\, Len(pq')$
PROVE $i + 1 \in 1 \,..\, Len(pq) \land pq'[i] = pq[i + 1]$

BY ⟨3⟩1, ⟨4⟩2

⟨5⟩.QED BY ⟨3⟩2, ⟨5⟩1, ⟨5⟩2 DEF *ClockInvInner*, *beats*

⟨4⟩3.CASE $\{p, q\} \neq \{n, k\}$

BY ⟨3⟩1, ⟨3⟩2, ⟨4⟩3 DEF *ClockInvInner*, *beats*

⟨4⟩.QED BY ⟨4⟩1, ⟨4⟩2, ⟨4⟩3, *Zenon*

$\langle 3 \rangle$.QED BY $\langle 3 \rangle 2$ DEF *ClockInv*

$\langle 2 \rangle 6$. ASSUME NEW $n \in Proc$, NEW $k \in Proc \setminus \{n\}$, $ReceiveRelease(n, k)$
PROVE $ClockInv'$

$\langle 3 \rangle 1$. $\land\ network[k][n] \neq \langle\rangle$
$\land\ Head(network[k][n]).type = \text{“rel”}$
$\land\ req' = [req \text{ EXCEPT } ![n][k] = 0]$
$\land\ network' = [network \text{ EXCEPT } ![k][n] = Tail(@)]$
$\land$ UNCHANGED $\langle clock, ack, crit \rangle$
$\land\ Contains(network[k][n], \text{“rel”})$

BY $\langle 2 \rangle 6$ DEF *ReceiveRelease*, *Contains* , *BasicInv*, *CommInv*, *Contains*

$\langle 3 \rangle 2$. $\land\ \neg Contains(network[n][k], \text{“ack”})$
$\land\ n \notin ack[k]$

BY $\langle 3 \rangle 1$, *Zenon* DEF *BasicInv*, *CommInv*

$\langle 3 \rangle 3$. ASSUME NEW $p \in Proc$, NEW $q \in Proc$, $ClockInvInner(p, q)$
PROVE $ClockInvInner(p, q)'$

$\langle 4 \rangle$.DEFINE $pq \triangleq network[p][q]$
$qp \triangleq network[q][p]$

$\langle 4 \rangle 1$.CASE $p = n \land q = k$

$\langle 5 \rangle$. $\land$ UNCHANGED $\langle pq, ack, req[p][p], req[q][p], clock \rangle$
$\land\ beats(p, q)'$
$\land\ \forall\, i \in 1 \,..\, Len(qp') : i + 1 \in 1 \,..\, Len(qp) \land qp'[i] = qp[i + 1]$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$ DEF *beats*

$\langle 5 \rangle$.$Contains(qp', \text{“ack”}) \equiv Contains(qp, \text{“ack”})$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$, *ContainsTail* DEF *BasicInv*, *NetworkInv*

$\langle 5 \rangle$.$Precedes(qp', \text{“ack”}, \text{“req”}) \Rightarrow Precedes(qp, \text{“ack”}, \text{“req”})$

BY $\langle 3 \rangle 1$, $\langle 4 \rangle 1$, *PrecedesInTail*, *Zenon*

$\langle 5 \rangle$.QED BY $\langle 3 \rangle 3$, *Zenon* DEF *ClockInvInner*

$\langle 4 \rangle 2$.CASE $p = k \land q = n$

$\langle 5 \rangle$. $\land$ UNCHANGED $\langle qp, ack, req[p][p], req[p][q], crit, clock \rangle$
$\land\ \neg Contains(qp', \text{“ack”}) \land q \notin ack'[p]$
$\land\ \forall\, i \in 1 \,..\, Len(pq') : i + 1 \in 1 \,..\, Len(pq) \land pq'[i] = pq[i + 1]$

BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 4 \rangle 2$

$\langle 5 \rangle$.QED BY $\langle 3 \rangle 3$ DEF *ClockInvInner*, *beats*

$\langle 4 \rangle 3$.CASE $\{p, q\} \neq \{k, n\}$

BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$, $\langle 4 \rangle 3$ DEF *ClockInvInner*, *beats*

$\langle 4 \rangle$.QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$, *Zenon*

$\langle 3 \rangle$.QED BY $\langle 3 \rangle 3$ DEF *ClockInv*

$\langle 2 \rangle 7$.CASE UNCHANGED *vars*

BY $\langle 2 \rangle 7$ DEF *ClockInv*, *ClockInvInner*, *beats*, *vars*

$\langle 2 \rangle 8$. QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$ DEF *Next*

$\langle 1 \rangle$.QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, *TypeCorrect*, *BasicInvariant*, *PTL* DEF *Spec*

Mutual exclusion is a simple consequence of the above invariants. In particular, if two distinct processes p and q were ever in the critical section at the same instant, then $beats(p, q)$ and $beats(q, p)$ would both have to hold, but this is impossible.

THEOREM *Safety* $\triangleq$ *Spec* $\Rightarrow \Box$*Mutex*

$\langle 1 \rangle 1.$ *TypeOK* $\wedge$ *BasicInv* $\wedge$ *ClockInv* $\Rightarrow$ *Mutex*

  $\langle 2 \rangle$.SUFFICES ASSUME *TypeOK*, *BasicInv*, *ClockInv*,
                 NEW $p \in crit$, NEW $q \in crit$, $p \neq q$
         PROVE FALSE

   BY DEF *Mutex*

  $\langle 2 \rangle$.USE DEF *TypeOK*

  $\langle 2 \rangle$. $\wedge\ req[p][p] > 0 \wedge req[q][q] > 0$
     $\wedge\ p \in ack[q] \wedge q \in ack[p]$

   BY DEF *BasicInv*, *CommInv*

  $\langle 2 \rangle$. $\wedge\ req[q][p] = req[p][p]$
     $\wedge\ req[p][q] = req[q][q]$
     $\wedge\ beats(p, q)$
     $\wedge\ beats(q, p)$

   BY DEF *ClockInv*, *ClockInvInner*

  $\langle 2 \rangle$.QED BY *NType* DEF *Proc*, *beats*

$\langle 1 \rangle$.QED BY *TypeCorrect*, *BasicInvariant*, *ClockInvariant*, $\langle 1 \rangle 1$, *PTL*