MODULE *PlusCalLIFO*

TLA+

```
EXTENDS Naturals, Sequences, TLC
CONSTANT QueueSize, Message
```

PlusCal

```
--algorithm FIFO{
  variables
    inchan ∈ [val : Message, rdy : {0, 1}, ack : {0, 1}],

    outchan ∈ [val : Message, rdy : {0, 1}, ack : {0, 1}] ;

    q = ⟨⟩ ;
```

Added a *queueSize* constant to the typeinvariants, so that the buffer had a finite number of elements

```
  macro CheckInvariants( chan ) {
    assert (chan.val ∈ Message) ;
    assert (chan.rdy ∈ {0, 1}) ;
    assert (chan.ack ∈ {0, 1}) ;
    assert (q ∈ Seq(Message)) ;
    assert (QueueSize > 0) ;
    assert (QueueSize ∈ Nat) ;
    assert (Len(q) ≤ QueueSize) ;
   }
```

Changed sender send to only be applicable when the current length of the queue is lower than the maximum queue size

```
  process ( SSend = "ssend" )
    variable oldrdy ;
  {
    ss0:  while ( TRUE ) {
    ss1:    await inchan.rdy = inchan.ack ;
    ss2:    oldrdy := inchan.rdy ;
            inchan.rdy := 1 − inchan.rdy ;
            CheckInvariants(inchan) ;
            assert (inchan.rdy ≠ oldrdy) ;
            assert (inchan.rdy ≠ inchan.ack) ;
           }

   } ;   end process SSend
```

Receive message from channel *in* . change the queue to contain a concatination of the new value from the in channel and the original queue

```
  process ( BufRcv = "bufrcv" )
        variable oldack ;
  {
    $br0$:  while ( TRUE ) {
    $br1$:      await $inchan.rdy \neq inchan.ack \land Len(q) < QueueSize$ ;
    $br2$:      $oldack := inchan.ack$ ;
                $inchan.ack := 1 - inchan.ack$ ;
                $q := \langle inchan.val \rangle \circ q$ ;
                $CheckInvariants(inchan)$ ;
                assert $(inchan.ack \neq oldack)$ ;
                assert $(inchan.rdy = inchan.ack)$ ;
              }
  } ;   end process $BufRecv$

process ( $BufSend$ = "bufsend" )
    variable $oldrdy$, $rval$ ;
{
    $bs0$:  while ( TRUE ) {
    $bs1$:      await $outchan.rdy = outchan.ack \land q \neq \langle\rangle$ ;
    $bs2$:      $oldrdy := outchan.rdy$ ;
                $outchan.rdy := 1 - outchan.rdy$ ;
                $rval := Head(q)$ ;
                $q := Tail(q)$ ;

                $CheckInvariants(outchan)$ ;
                assert $(outchan.rdy \neq oldrdy)$ ;
                assert $(outchan.rdy \neq outchan.ack)$ ;


    $bs3$:      $outchan.val := rval$ ;
    Hack to get value into $outchan$. Not able to do it in $bs2$. $outchan.val := Head(q)$ requires its
    own label and therefor couldn't be done in $bs2$.
              }

  } ;   end process $BufSend$

process ( $RRcv$ = "rrcv" )
        variable $oldack$ ;
{
    $rr0$:  while ( TRUE ) {
    $rr1$:      await $outchan.rdy \neq outchan.ack$ ;
    $rr2$:      $oldack := outchan.ack$ ;
                $outchan.ack := 1 - outchan.ack$ ;
                $CheckInvariants(outchan)$ ;
                assert $(outchan.ack \neq oldack)$ ;
                assert $(outchan.rdy = outchan.ack)$ ;
              }
  } ;   end process $RRecv$

}
```

BEGIN TRANSLATION

Process variable *oldrdy* of process *SSend* at line 33 col 14 changed to *oldrdy_*

Process variable *oldack* of process *BufRcv* at line 51 col 18 changed to *oldack_*

$\textsc{constant}\ \mathit{defaultInitValue}$

$\textsc{variables}\ \mathit{inchan},\ \mathit{outchan},\ q,\ \mathit{pc},\ \mathit{oldrdy\_},\ \mathit{oldack\_},\ \mathit{oldrdy},\ \mathit{rval},\ \mathit{oldack}$

$$\mathit{vars} \triangleq \langle \mathit{inchan},\ \mathit{outchan},\ q,\ \mathit{pc},\ \mathit{oldrdy\_},\ \mathit{oldack\_},\ \mathit{oldrdy},\ \mathit{rval},\ \mathit{oldack} \rangle$$

$$\mathit{ProcSet} \triangleq \{\text{"ssend"}\} \cup \{\text{"bufrcv"}\} \cup \{\text{"bufsend"}\} \cup \{\text{"rrcv"}\}$$

$\mathit{Init} \triangleq$ Global variables

$$\begin{aligned}
&\land\ \mathit{inchan} \in [\mathit{val} : \mathit{Message},\ \mathit{rdy} : \{0, 1\},\ \mathit{ack} : \{0, 1\}] \\
&\land\ \mathit{outchan} \in [\mathit{val} : \mathit{Message},\ \mathit{rdy} : \{0, 1\},\ \mathit{ack} : \{0, 1\}] \\
&\land\ q = \langle\rangle
\end{aligned}$$

Process *SSend*

$$\land\ \mathit{oldrdy\_} = \mathit{defaultInitValue}$$

Process *BufRcv*

$$\land\ \mathit{oldack\_} = \mathit{defaultInitValue}$$

Process *BufSend*

$$\begin{aligned}
&\land\ \mathit{oldrdy} = \mathit{defaultInitValue} \\
&\land\ \mathit{rval} = \mathit{defaultInitValue}
\end{aligned}$$

Process *RRcv*

$$\begin{aligned}
&\land\ \mathit{oldack} = \mathit{defaultInitValue} \\
&\land\ \mathit{pc} = [\mathit{self} \in \mathit{ProcSet} \mapsto \textsc{case}\ \mathit{self} = \text{"ssend"} \to \text{"ss0"} \\
&\qquad\qquad\qquad\qquad\qquad\quad \Box\ \ \mathit{self} = \text{"bufrcv"} \to \text{"br0"} \\
&\qquad\qquad\qquad\qquad\qquad\quad \Box\ \ \mathit{self} = \text{"bufsend"} \to \text{"bs0"} \\
&\qquad\qquad\qquad\qquad\qquad\quad \Box\ \ \mathit{self} = \text{"rrcv"} \to \text{"rr0"}]
\end{aligned}$$

$$\begin{aligned}
\mathit{ss}0 \triangleq\ &\land\ \mathit{pc}[\text{"ssend"}] = \text{"ss0"} \\
&\land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{"ssend"}] = \text{"ss1"}] \\
&\land\ \textsc{unchanged}\ \langle \mathit{inchan},\ \mathit{outchan},\ q,\ \mathit{oldrdy\_},\ \mathit{oldack\_},\ \mathit{oldrdy},\ \mathit{rval}, \\
&\qquad\qquad\qquad\quad \mathit{oldack} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathit{ss}1 \triangleq\ &\land\ \mathit{pc}[\text{"ssend"}] = \text{"ss1"} \\
&\land\ \mathit{inchan.rdy} = \mathit{inchan.ack} \\
&\land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{"ssend"}] = \text{"ss2"}] \\
&\land\ \textsc{unchanged}\ \langle \mathit{inchan},\ \mathit{outchan},\ q,\ \mathit{oldrdy\_},\ \mathit{oldack\_},\ \mathit{oldrdy},\ \mathit{rval}, \\
&\qquad\qquad\qquad\quad \mathit{oldack} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathit{ss}2 \triangleq\ &\land\ \mathit{pc}[\text{"ssend"}] = \text{"ss2"} \\
&\land\ \mathit{oldrdy\_}' = \mathit{inchan.rdy} \\
&\land\ \mathit{inchan}' = [\mathit{inchan}\ \textsc{except}\ !.\mathit{rdy} = 1 - \mathit{inchan.rdy}] \\
&\land\ \mathit{Assert}((\mathit{inchan}'.\mathit{val} \in \mathit{Message}), \\
&\qquad\qquad \text{"Failure of assertion at line 20, column 5 of macro called at line 39, column 13."}) \\
&\land\ \mathit{Assert}((\mathit{inchan}'.\mathit{rdy} \in \{0, 1\}), \\
&\qquad\qquad \text{"Failure of assertion at line 21, column 5 of macro called at line 39, column 13."}) \\
&\land\ \mathit{Assert}((\mathit{inchan}'.\mathit{ack} \in \{0, 1\}),
\end{aligned}$$

"Failure of assertion at line 22, column 5 of macro called at line 39, column 13.")
$\land\ Assert((q \in Seq(Message)),$
"Failure of assertion at line 23, column 5 of macro called at line 39, column 13.")
$\land\ Assert((QueueSize > 0),$
"Failure of assertion at line 24, column 5 of macro called at line 39, column 13.")
$\land\ Assert((QueueSize \in Nat),$
"Failure of assertion at line 25, column 5 of macro called at line 39, column 13.")
$\land\ Assert((Len(q) \leq QueueSize),$
"Failure of assertion at line 26, column 5 of macro called at line 39, column 13.")
$\land\ Assert((inchan'.rdy \neq oldrdy\_'),$
"Failure of assertion at line 40, column 13.")
$\land\ Assert((inchan'.rdy \neq inchan'.ack),$
"Failure of assertion at line 41, column 13.")
$\land\ pc' = [pc \text{ EXCEPT } ![\text{"ssend"}] = \text{"ss0"}]$
$\land\ \text{UNCHANGED } \langle outchan, q, oldack\_, oldrdy, rval, oldack\rangle$

$SSend \triangleq ss0 \lor ss1 \lor ss2$

$br0 \triangleq \land\ pc[\text{"bufrcv"}] = \text{"br0"}$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{"bufrcv"}] = \text{"br1"}]$
$\land\ \text{UNCHANGED } \langle inchan, outchan, q, oldrdy\_, oldack\_, oldrdy, rval,$
$oldack\rangle$

$br1 \triangleq \land\ pc[\text{"bufrcv"}] = \text{"br1"}$
$\land\ inchan.rdy \neq inchan.ack \land Len(q) < QueueSize$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{"bufrcv"}] = \text{"br2"}]$
$\land\ \text{UNCHANGED } \langle inchan, outchan, q, oldrdy\_, oldack\_, oldrdy, rval,$
$oldack\rangle$

$br2 \triangleq \land\ pc[\text{"bufrcv"}] = \text{"br2"}$
$\land\ oldack\_' = inchan.ack$
$\land\ inchan' = [inchan \text{ EXCEPT } !.ack = 1 - inchan.ack]$
$\land\ q' = \langle inchan'.val\rangle \circ q$
$\land\ Assert((inchan'.val \in Message),$
"Failure of assertion at line 20, column 5 of macro called at line 58, column 13.")
$\land\ Assert((inchan'.rdy \in \{0, 1\}),$
"Failure of assertion at line 21, column 5 of macro called at line 58, column 13.")
$\land\ Assert((inchan'.ack \in \{0, 1\}),$
"Failure of assertion at line 22, column 5 of macro called at line 58, column 13.")
$\land\ Assert((q' \in Seq(Message)),$
"Failure of assertion at line 23, column 5 of macro called at line 58, column 13.")
$\land\ Assert((QueueSize > 0),$
"Failure of assertion at line 24, column 5 of macro called at line 58, column 13.")
$\land\ Assert((QueueSize \in Nat),$
"Failure of assertion at line 25, column 5 of macro called at line 58, column 13.")
$\land\ Assert((Len(q') \leq QueueSize),$

"Failure of assertion at line 26, column 5 of macro called at line 58, column 13.")
$\land$ $Assert((inchan'.ack \neq oldack\_'),$
"Failure of assertion at line 59, column 13.")
$\land$ $Assert((inchan'.rdy = inchan'.ack),$
"Failure of assertion at line 60, column 13.")
$\land$ $pc' = [pc \text{ EXCEPT } ![\text{"bufrcv"}] = \text{"br0"}]$
$\land$ $\text{UNCHANGED } \langle outchan, oldrdy\_, oldrdy, rval, oldack \rangle$

$BufRcv \triangleq br0 \lor br1 \lor br2$

$bs0 \triangleq$ $\land$ $pc[\text{"bufsend"}] = \text{"bs0"}$
$\land$ $pc' = [pc \text{ EXCEPT } ![\text{"bufsend"}] = \text{"bs1"}]$
$\land$ $\text{UNCHANGED } \langle inchan, outchan, q, oldrdy\_, oldack\_, oldrdy, rval,$
$oldack \rangle$

$bs1 \triangleq$ $\land$ $pc[\text{"bufsend"}] = \text{"bs1"}$
$\land$ $outchan.rdy = outchan.ack \;\; \land q \neq \langle\rangle$
$\land$ $pc' = [pc \text{ EXCEPT } ![\text{"bufsend"}] = \text{"bs2"}]$
$\land$ $\text{UNCHANGED } \langle inchan, outchan, q, oldrdy\_, oldack\_, oldrdy, rval,$
$oldack \rangle$

$bs2 \triangleq$ $\land$ $pc[\text{"bufsend"}] = \text{"bs2"}$
$\land$ $oldrdy' = outchan.rdy$
$\land$ $outchan' = [outchan \text{ EXCEPT } !.rdy = 1 - outchan.rdy]$
$\land$ $rval' = Head(q)$
$\land$ $q' = Tail(q)$
$\land$ $Assert((outchan'.val \in Message),$
"Failure of assertion at line 20, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((outchan'.rdy \in \{0, 1\}),$
"Failure of assertion at line 21, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((outchan'.ack \in \{0, 1\}),$
"Failure of assertion at line 22, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((q' \in Seq(Message)),$
"Failure of assertion at line 23, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((QueueSize > 0),$
"Failure of assertion at line 24, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((QueueSize \in Nat),$
"Failure of assertion at line 25, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((Len(q') \leq QueueSize),$
"Failure of assertion at line 26, column 5 of macro called at line 74, column 13.")
$\land$ $Assert((outchan'.rdy \neq oldrdy'),$
"Failure of assertion at line 75, column 13.")
$\land$ $Assert((outchan'.rdy \neq outchan'.ack),$
"Failure of assertion at line 76, column 13.")
$\land$ $pc' = [pc \text{ EXCEPT } ![\text{"bufsend"}] = \text{"bs3"}]$
$\land$ $\text{UNCHANGED } \langle inchan, oldrdy\_, oldack\_, oldack \rangle$

$$bs3 \triangleq \begin{array}[t]{l} \land\, pc[\text{“bufsend”}] = \text{“bs3”} \\ \land\, outchan' = [outchan \text{ EXCEPT } !.val = rval] \\ \land\, pc' = [pc \text{ EXCEPT } ![\text{“bufsend”}] = \text{“bs0”}] \\ \land\, \text{UNCHANGED } \langle inchan,\ q,\ oldrdy\_,\ oldack\_,\ oldrdy,\ rval,\ oldack \rangle \end{array}$$

$$BufSend \triangleq bs0 \lor bs1 \lor bs2 \lor bs3$$

$$rr0 \triangleq \begin{array}[t]{l} \land\, pc[\text{“rrcv”}] = \text{“rr0”} \\ \land\, pc' = [pc \text{ EXCEPT } ![\text{“rrcv”}] = \text{“rr1”}] \\ \land\, \text{UNCHANGED } \langle inchan,\ outchan,\ q,\ oldrdy\_,\ oldack\_,\ oldrdy,\ rval, \\ \qquad\qquad\qquad oldack \rangle \end{array}$$

$$rr1 \triangleq \begin{array}[t]{l} \land\, pc[\text{“rrcv”}] = \text{“rr1”} \\ \land\, outchan.rdy \neq outchan.ack \\ \land\, pc' = [pc \text{ EXCEPT } ![\text{“rrcv”}] = \text{“rr2”}] \\ \land\, \text{UNCHANGED } \langle inchan,\ outchan,\ q,\ oldrdy\_,\ oldack\_,\ oldrdy,\ rval, \\ \qquad\qquad\qquad oldack \rangle \end{array}$$

$$rr2 \triangleq \begin{array}[t]{l} \land\, pc[\text{“rrcv”}] = \text{“rr2”} \\ \land\, oldack' = outchan.ack \\ \land\, outchan' = [outchan \text{ EXCEPT } !.ack = 1 - outchan.ack] \\ \land\, Assert((outchan'.val \in Message), \\ \qquad\qquad \text{“Failure of assertion at line 20, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((outchan'.rdy \in \{0, 1\}), \\ \qquad\qquad \text{“Failure of assertion at line 21, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((outchan'.ack \in \{0, 1\}), \\ \qquad\qquad \text{“Failure of assertion at line 22, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((q \in Seq(Message)), \\ \qquad\qquad \text{“Failure of assertion at line 23, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((QueueSize > 0), \\ \qquad\qquad \text{“Failure of assertion at line 24, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((QueueSize \in Nat), \\ \qquad\qquad \text{“Failure of assertion at line 25, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((Len(q) \leq QueueSize), \\ \qquad\qquad \text{“Failure of assertion at line 26, column 5 of macro called at line 95, column 13.”}) \\ \land\, Assert((outchan'.ack \neq oldack'), \\ \qquad\qquad \text{“Failure of assertion at line 96, column 13.”}) \\ \land\, Assert((outchan'.rdy = outchan'.ack), \\ \qquad\qquad \text{“Failure of assertion at line 97, column 13.”}) \\ \land\, pc' = [pc \text{ EXCEPT } ![\text{“rrcv”}] = \text{“rr0”}] \\ \land\, \text{UNCHANGED } \langle inchan,\ q,\ oldrdy\_,\ oldack\_,\ oldrdy,\ rval \rangle \end{array}$$

$$RRcv \triangleq rr0 \lor rr1 \lor rr2$$

$$Next \triangleq SSend \lor BufRcv \lor BufSend \lor RRcv$$

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

```
END TRANSLATION
```

\* Modification History
\* Last modified *Wed Mar* 07 11:38:01 *CET* 2018 by *jacob*
\* Created *Thu Mar* 01 11:47:28 *CET* 2018 by *jacob*