Safely Sliding Windows

Dirk Henkel
RVS Group Research Report RVS-RR-97-05a
Universität Bielefeld

5 May, 1997
last changed 19 November, 1997

Abstract

A formal TLA proof is given that the sliding window message retransmission mechanism with linear arithmetic implements the safety property of a reliable, lossless, asynchronous FIFO queue. In order to make the verification as routine as possible, the proof is divided into four parts with three additional specifications intermediate between the requirements specification and the sliding window specification.

Introduction

This paper provides the safety part of a verification proof for a sliding-window-style message retransmission mechanism[1] with linear arithmetic (SWLA). SWLA is the linear arithmetic version of the sliding window message retransmission mechanism with modular arithmetic (SWMA), which is part of the Radio Link Protocol (RLP1), the datalink layer protocol for TDMA mobile cellular radio being analysed by the FCAR project of the Telecommunications Software Group at INRS-Télécommunications in Montréal [Fer96]. The specifications and proofs in this paper contribute to the verification of SWLA using TLA and TLA$^+$.

Documents on the Temporal Logic of Actions (TLA) calculus and the corresponding specification language TLA$^+$ can be found at the TLA website of Leslie Lamport [Lam]. A tutorial on TLA verification has been written by Peter Ladkin [Lad97].

TLA provides a clear separation between safety and liveness properties. The following specifications and proofs concern only safety properties. To obtain complete specifications we must add liveness properties, and to obtain a complete verification of SWLA, we ought to prove its liveness properties, too. Liveness properties in TLA specifications have the form of Boolean combinations of fairness properties. The specifications have been designed, so to speak, with these liveness properties in mind, so we don't believe that adding them will change the general idea of how the five specified systems work. The only sure way to know, however, would be to add them and provide the relevant proofs.

System verification must be performed relative to stated requirements. For SWLA (as well as for RLP1) we require that the protocol implements a reliable, asynchronous first-in first-out message-transfer stream. We specify this by means of a variable *buffer* which contains the messages that have been *sent* but not *received*, in order. Reliability is ensured by the implicit criterion that sent messages do not disappear; asynchrony by the independence of *send* and *receive* actions (corresponding to the *buffer*'s *push* and *pop* actions); FIFO by the standard FIFO property of a buffer. The *AbstractBuffer* (AB) is thus a state

[1]It has been pointed out by Michael Ferguson that the meaning of the word "protocol" in telecommunications engeneering is far richer than its informal use by specification engineers and network textbook writers. We find the mathematical operation far better described as a "retransmission mechanism", as Ferguson suggests.

machine which embodies exactly the requirements of the communication system. The AB specification in TLA was first formulated and used in [Lad96].

For both SWLA and AB we give formal specifications of their safety properties: the TLA formulas $I.\mathit{Safety}$ and $V.\mathit{Safety}$. Proving that SWLA-safety implies AB-safety is equivalent to proving that $I.\mathit{Safety}$ logically implies $V.\mathit{Safety}$. One may try this directly, but it's easier to do it in several small steps via a hierarchy of specifications. We use three intermediate specifications $II.\mathit{Safety}$, $III.\mathit{Safety}$ and $IV.\mathit{Safety}$, which show a continuous development from the simplicity of $I.\mathit{Safety}$ to the details of $V.\mathit{Safety}$. The whole safety proof is accomplished in four steps of the form "$(N+1).\mathit{Safety}$ implies $N.\mathit{Safety}$". A summary of the five TLA^+ specifications without the proofs is given in [GHL97].

The approach to the system verification differs from that of Mokkedem, Ferguson and Johnston [MFJ97]. They describe invariants, and prove using the TLA logic that these invariants logically follow from an implementation specified in TLA. We follow the "traditional" TLA format, in which all specifications, *including the requirement*, have the form of a state machine. In this example, the requirement state machine embodies, as we have argued, exactly the properties of a reliable, asynchronous FIFO channel. It is notable that the required invariant in [MFJ97] does not appear to be logically equivalent to a TLA specification as defined by Lamport [Lam94], thus it is not guaranteed that the TLA logic suffices to prove it (but Mokkedem, Ferguson and Johnston have indeed succeeded in proving it).

The five safety specifications are given in six TLA^+ modules. The first one is a collection of some common definitions. It is EXTEND'd by each of the five specification modules I to V. Module I, the "requirement", specifies the safety properties of AB, which is just a sequence with fixed maximum length and the two actions *Push* and *Pop*. As we said above, the purpose of any reliable, asynchronous FIFO message-transmission mechanism must be to simulate this precisely. Module *II* is a translation of AB into set notation. Module *III* introduces a distinction between sender and receiver sides. In Module *IV* the message transfer from sender to receiver is split into two seperate actions. Module *V* specifies the safety properties of SWLA. Besides *Push* and *Pop*, it defines four actions that make up the message transfer. Each module is preceeded by some commentary that should give an idea of how the specified system works.

Section 1 introduces the common definitions module. In section 2 the AB-safety requirement is specified by module I. Each of the sections 3 to 6 contains another specification module, a TLA^+ module stating that this specification implies the previous one, and the corresponding TLA proof.

Acknowledgements

The specifications were developed jointly with Thorsten Gerdsmeier, Prof. Dr. Jean-Charles Grégoire of INRS-Télécommunications à l'Université de Québec in Montréal and Prof. Dr. Peter Ladkin. Petra Korff also contributed to the analysis and verification of some of the former high-level specifications with which we worked. Some of the explanatory commentary to the specifications is based on notes by Prof. Dr. Grégoire.

References

[Fer] Michael J. Ferguson: Selected publications. Available through `http://www.inrs-telecom.uquebec.ca/users/telesoft/`

[Fer96] Michael J. Ferguson: Formalization and validation of the Radio Link Protocol (RLP1). To be published in *Computer Networks and ISDN Systems*, 1996. Available through [Fer].

[GHL97] Thorsten Gerdsmeier, Dirk Henkel, Peter B. Ladkin: Safely sliding windows: the specs. Technical Report RVS-RR-97-05b, RVS Group, Universität Bielefeld, Technische Fakultät, July 1997. Available through [Lad].

[Lad] Peter B. Ladkin: Abstracts of RVS research publications. Available through `http://www.rvs.uni-bielefeld.de`

[Lad96] Peter B. Ladkin: Formal but lively buffers in TLA^+. Technical Report RVS-RR-96-07, RVS Group, Universität Bielefeld, Technische Fakultät, January 1996. Available through [Lad].

[Lad97] Peter B. Ladkin: Using the Temporal Logic of Actions: A tutorial on TLA verification. Technical Report RVS-RR-97-08, RVS Group, Universität Bielefeld, Technische Fakultät, June 1997. Available through [Lad].

[Lam] Leslie Lamport: TLA – The Temporal Logic of Actions. `http://www.research.digital.com/SRC/tla/`

[Lam94] Leslie Lamport: The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872-923, May 1994. Also available through [Lam].

[MFJ97] Abdelillah Mokkedem, Michael J. Ferguson, and Robert deB. Johnston: A TLA solution to the specification and verification of the RLP1 retransmission protocol. *Formal Methods Europe* symposium, September 1997. Available through [Fer].

[Gre97] Jean-Charles Gregoire: TLA + PROMELA: conjecture, check, proof – engineering new protocols using methods and formal notations. *Formal Methods Europe* symposium, September 1997.

1 Common Definitions

Some declarations, assumptions and definitions will be needed in several of the following specifications. They are collected in the module *CommonDefinitions*.

Messages is the Set of all Objects that can be passed through the buffer we specify. It specifically does not contain the constant $\perp_{Messages}$. *BufferLength* is the maximum number of such objects that can be held in the buffer at any time. We require $Bufferlength \geq 1$.

We introduce a simple notation for intervals of natural numbers and some operators on sequences: $Len(s)$ denotes the length of a sequence s; $Tail(s)$ is "s without its first element"; $s \circ t$ the concatenation of the sequences s and t.

The last four definitions refer to sets of pairs of the form $(index, message)$ where *index* is a natural number. $MinPair(S)$ is that pair (or one of those pairs) with the least *index* in S, $MinIndex(S)$ is this least index. Similarly for *Max*-functions with the greatest index. These functions are defined only on nonempty sets.

module *CommonDefinitions*

EXTENDS *Naturals*

CONSTANTS $BufferLength, Messages, \perp_{Messages}$
ASSUMPTION $BufferLength > 0$
ASSUMPTION $\perp_{Messages} \notin Messages$

$$l \,..\, u \;\triangleq\; \{i \in Nat : l \leq i \wedge i \leq u\}$$
$$Len(s) \;\triangleq\; \text{CHOOSE } n : \begin{array}{l} \wedge\; n \in Nat \\ \wedge\; \text{DOMAIN } s = 1 \,..\, n \end{array}$$
$$Tail(s) \;\triangleq\; [i \in 1 \,..\, (Len(s) - 1) \mapsto s[i+1]]$$
$$s \circ t \;\triangleq\; [i \in 1 \,..\, (Len(s) + Len(t)) \mapsto \textbf{if } i \leq Len(s) \;\; \begin{array}{l} \textbf{then } s[i] \\ \textbf{else } \; t[i - Len(s)]] \end{array}$$
$$MinPair(S) \;\triangleq\; \text{CHOOSE } z : \begin{array}{l} \wedge\; z \in S \\ \wedge\; \forall\, y \in S : z[1] \leq y[1] \end{array}$$
$$MaxPair(S) \;\triangleq\; \text{CHOOSE } z : \begin{array}{l} \wedge\; z \in S \\ \wedge\; \forall\, y \in S : z[1] \geq y[1] \end{array}$$
$$MinIndex(S) \;\triangleq\; MinPair(S)[1]$$
$$MaxIndex(S) \;\triangleq\; MaxPair(S)[1]$$

2 An Abstract Buffer

Module I contains a straight-forward specification of an abstract buffer. Its only variable is the sequence *buffer*. At the beginning, *buffer* is empty. If the buffer is at any time not full (i.e. shorter than *BufferLength*), a message can be *Push*'d onto it at the rear end. If the buffer is not empty, its first message can be *Pop*'d off. Thus the messages are *Pop*'d in FIFO order.

module I

EXTENDS *CommonDefinitions*

VARIABLES *buffer*

$$Init \triangleq buffer = \langle\,\rangle$$

$$Push(a) \triangleq \begin{array}{l} \land\ a \in Messages \\ \land\ Len(buffer) < BufferLength \\ \land\ buffer' = buffer \circ \langle\, a \,\rangle \end{array}$$

$$Pop \triangleq \begin{array}{l} \land\ buffer \neq \langle\,\rangle \\ \land\ buffer' = Tail(buffer) \end{array}$$

$$SomeAction \triangleq \begin{array}{l} \lor\ \exists\, a : Push(a) \\ \lor\ Pop \end{array}$$

$$Safety \triangleq \begin{array}{l} \land\ Init \\ \land\ \Box[SomeAction]_{buffer} \end{array}$$

3 Set Notation

Specification II is a translation of the buffer into set notation. Instead of the sequence *buffer* we now use the set *tSet*. Like *buffer* in specification I, *tSet* starts empty.

To preserve the FIFO property of the buffer, we must know in which order the messages in the buffer have been *Push*'d. Thus $Push(a)$ does not simply add the message a to *tSet* but a pair $(index, a)$ where *index* is the current number of the *Push* action. Accordingly, *Pop* takes the elements out of *tSet* in ascending order of *index*.

The current indices for the *Push* and *Pop* actions are stored in the counter variables $index_s$ and $index_r$. Both start at 0; every *Push* increments $index_s$; every *Pop* increments $index_r$. Thus $index_s$ is the index of the next element to be *Push*'d and $index_r$ is the index of the next element to be *Pop*'d.

The difference between $index_s$ and $index_r$ equals the size of *tSet*, thus it can be used to specify that the buffer must not be full at the beginning of a *Push*.

module II

EXTENDS *CommonDefinitions*

VARIABLES $tSet, index_r, index_s$

$$
\begin{array}{lcl}
Vars & \triangleq & \langle tSet, index_s, index_r \rangle \\
Init & \triangleq & \land\ tSet = \emptyset \\
 & & \land\ index_s = 0 \\
 & & \land\ index_r = 0
\end{array}
$$

$$
\begin{array}{lcl}
Push(a) & \triangleq & \land\ a \in Messages \\
 & & \land\ index_s - index_r < BufferLength \\
 & & \land\ tSet' = tSet \cup \{\langle index_s, a \rangle\} \\
 & & \land\ index_s' = index_s + 1 \\
 & & \land\ \text{UNCHANGED } index_r \\
Pop & \triangleq & \land\ \exists\, a : \land\ \langle index_r, a \rangle \in tSet \\
 & & \qquad\quad\ \land\ tSet' = tSet \setminus \{\langle index_r, a \rangle\} \\
 & & \land\ index_r' = index_r + 1 \\
 & & \land\ \text{UNCHANGED } index_s
\end{array}
$$

$$
\begin{array}{lcl}
SomeAction & \triangleq & \lor\ \exists\, a : Push(a) \\
 & & \lor\ Pop \\
Safety & \triangleq & \land\ Init \\
 & & \land\ \Box[SomeAction]_{Vars}
\end{array}
$$

II.Safety implies I.Safety

module *II implies I*

EXTENDS *II*

$HLS \triangleq$ INSTANCE I WITH
$\quad buffer \leftarrow [i \in 1 \,..\, (index_s - index_r)$
$\qquad \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet]$

$f \sqcup g \triangleq [x \in (\text{DOMAIN } f \cup \text{DOMAIN } g) \mapsto \text{CHOOSE } y : y = f[x] \lor y = g[x]]$

THEOREM $Safety \Rightarrow HLS.Safety$

$\langle 0 \rangle 1.$ $Safety \Rightarrow HLS.Safety$

PROOF:
$\langle 1 \rangle 1.$ $Init \Rightarrow HLS.Init$

PROOF:
$\langle 2 \rangle 1.$
$$\begin{aligned} &\land tSet = \emptyset \\ &\land index_s = 0 \\ &\land index_r = 0 \\ &\Rightarrow buffer = \langle\, \rangle \end{aligned}$$

PROOF: From $index_s = 0$, $index_r = 0$ and the definition of *buffer* follows
$$\begin{aligned} buffer &= [i \in 1 \,..\, 0 \mapsto \text{CHOOSE } x : \langle i - 1, x \rangle \in tSet] \\ &= [i \in \emptyset \mapsto \text{CHOOSE } x : \langle i - 1, x \rangle \in tSet] \\ &= \langle\, \rangle. \ \square \end{aligned}$$

$\langle 2 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of *Init* and *HLS.Init*. $\square$

$\langle 1 \rangle 2.$
$$\begin{aligned} &\land Init \\ &\land \Box[SomeAction]_{Vars} \\ &\Rightarrow \Box[HLS.SomeAction]_{buffer} \end{aligned}$$

PROOF: We use the following invariant:
LET: $Inv \triangleq$
$$\begin{aligned} 1.&\land \lor tSet = \emptyset \\ &\quad \lor MaxIndex(tSet) < index_s \\ 2.&\land |tSet| = index_s - index_r \\ 3.&\land \forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1 \end{aligned}$$

We use $Inv.1$ and $Inv.3$ to prove that *Push* simulates *HLS.Push*.
We use $Inv.1$, $Inv.2$ and $Inv.3$ to prove that *Pop* simulates *HLS.Pop*.
We use $Inv.1$ to prove the invariance of $Inv.2$ and $Inv.3$ under *Push*.

$\langle 2 \rangle 1.$
$$\begin{aligned} &\land Init \\ &\land \Box[SomeAction]_{Vars} \\ &\Rightarrow \Box Inv \end{aligned}$$

PROOF:

$\langle 3\rangle 1.$ $Init \Rightarrow Inv$

PROOF: From $index_s = 0$ and $index_r = 0$ follows $index_s - index_r = 0$.
$Inv.1$ follows from $tSet = \emptyset$.
$Inv.2$ follows from $tSet = \emptyset$ and $index_s - index_r = 0$.
$Inv.3$ follows from $index_s - index_r = 0$. $\square$

$\langle 3\rangle 2.$ $\wedge\ Inv$
$\quad\wedge\ \Box[SomeAction]_{Vars}$
$\quad\Rightarrow \Box Inv$

PROOF:

$\langle 4\rangle 1.$ $\wedge\ Inv$
$\quad\wedge\ [SomeAction]_{Vars}$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 5\rangle 1.$ $\wedge\ Inv$
$\quad\wedge\ \vee\ \exists\, a\ :\ Push(a)$
$\qquad\vee\ Pop$
$\qquad\vee\ \text{UNCHANGED}\ Vars$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 6\rangle 1.$ $\wedge\ Inv$
$\quad\wedge\ \exists\, a\ :\ Push(a)$
$\quad\Rightarrow Inv'$

PROOF:
LET: a : CONSTANT

$\langle 7\rangle 1.$ $\wedge\ Inv$
$\quad\wedge\ Push(a)$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 8\rangle 1.$ $\wedge\ Inv$
$\quad\wedge\ a \in Messages$
$\quad\wedge\ index_s - index_r < BufferLength$
$\quad\wedge\ tSet' = tSet \cup \{\langle index_s, a\rangle\}$
$\quad\wedge\ index'_s = index_s + 1$
$\quad\wedge\ \text{UNCHANGED}\ index_r$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 9\rangle 1.$ $\wedge\ \vee\ tSet = \emptyset$
$\qquad\vee\ MaxIndex(tSet) < index_s$
$\quad\wedge\ tSet' = tSet \cup \{\langle index_s, a\rangle\}$
$\quad\wedge\ index'_s = index_s + 1$

$\Rightarrow \lor\ tSet' = \emptyset$
$\quad \lor\ MaxIndex(tSet') < index'_s$

PROOF: From $tSet = \emptyset \lor MaxIndex(tSet) < index_s$ and $tSet' = tSet \cup \{\langle index_s, a\rangle\}$ follows $MaxIndex(tSet') = index_s$.
From this and $index'_s = index_s + 1$ follows $MaxIndex(tSet') < index'_s$. □

⟨9⟩2. $\land\ |tSet| = index_s - index_r$
$\land \lor\ tSet = \emptyset$
$\quad \lor\ MaxIndex(tSet) < index_s$
$\land\ tSet' = tSet \cup \{\langle index_s, a\rangle\}$
$\land\ index'_s = index_s + 1$
$\land$ UNCHANGED $index_r$
$\Rightarrow |tSet'| = index'_s - index'_r$

PROOF: From $tSet = \emptyset \lor MaxIndex(tSet) < index_s$ follows $\langle index_s, a\rangle \notin tSet$.
From this and $tSet' = tSet \cup \{\langle index_s, a\rangle\}$ follows $|tSet'| = |tSet| + 1$.
From $index'_s = index_s + 1$ and UNCHANGED $index_r$ follows
$index'_s - index'_r = index_s - index_r + 1$. Now we obtain the following equality:
$$\begin{aligned} |tSet'| &= |tSet| + 1 \\ &= index_s - index_r + 1 \\ &= index'_s - index'_r. \end{aligned}$$ □

⟨9⟩3. $\land\ \forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$
$\land \lor\ tSet = \emptyset$
$\quad \lor\ MaxIndex(tSet) < index_s$
$\land\ tSet' = tSet \cup \{\langle index_s, a\rangle\}$
$\land\ index'_s = index_s + 1$
$\land$ UNCHANGED $index_r$
$\Rightarrow \forall i \in 1 \,..\, (index'_s - index'_r) : |\{p \in tSet' : p[1] = i + index'_r - 1\}| = 1$

PROOF: From $tSet = \emptyset \lor MaxIndex(tSet) < index_s$
follows $|\{p \in tSet : p[1] = index_s\}| = 0$.
From this and $tSet' = tSet \cup \{\langle index_s, a\rangle\}$ follows $|\{p \in tSet' : p[1] = index_s\}| = 1$.
From this and $index'_s = index_s + 1$ follows $|\{p \in tSet' : p[1] = index'_s - 1\}| = 1$.
For all $i \in 1 \,..\, (index_s - index_r)$ holds $i + index_r - 1 < index_s$.
From this, $tSet' = tSet \cup \{\langle index_s, a\rangle\}$ and
$\forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$
follows $\forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet' : p[1] = i + index_r - 1\}| = 1$.
From this, $index'_s = index_s + 1$ and UNCHANGED $index_r$
follows $\forall i \in 1 \,..\, (index'_s - index'_r - 1) : |\{p \in tSet' : p[1] = i + index'_r - 1\}| = 1$.
From this and $|\{p \in tSet' : p[1] = index'_s - 1\}| = 1$
follows $\forall i \in 1 \,..\, (index'_s - index'_r) : |\{p \in tSet' : p[1] = i + index'_r - 1\}| = 1$. □

⟨9⟩4. Q.E.D.
PROOF: Propositional logic from ⟨9⟩1, ⟨9⟩2 and ⟨9⟩3. □

⟨8⟩2. Q.E.D.
PROOF: Direct from the definition of $Push(a)$. □

⟨7⟩2. Q.E.D.

PROOF: Predicate logic: forall-intro. □

$\langle 6 \rangle 2.$ $\wedge$ *Inv*
$\wedge$ *Pop*
$\Rightarrow$ *Inv'*

PROOF:
$\langle 7 \rangle 1.$ $\wedge\ Inv$
$\wedge\ \exists\, a : \wedge\ \langle index_r, a \rangle \in tSet$
$\wedge\ tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index_r' = index_r + 1$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow Inv'$

PROOF:
LET: a : CONSTANT

$\langle 8 \rangle 1.$ $\wedge\ Inv$
$\wedge\ \langle index_r, a \rangle \in tSet$
$\wedge\ tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index_r' = index_r + 1$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow Inv'$

PROOF:
$\langle 9 \rangle 1.$ $\wedge \vee\ tSet = \emptyset$
$\vee\ MaxIndex(tSet) < index_s$
$\wedge\ \langle index_r, a \rangle \in tSet$
$\wedge\ tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow \vee\ tSet' = \emptyset$
$\vee\ MaxIndex(tSet') < index_s'$

PROOF: From $tSet = \emptyset \vee MaxIndex(tSet) < index_s$ and $\langle index_r, a \rangle \in tSet$ follows $MaxIndex(tSet) < index_s$. From $tSet' = tSet \setminus \{\langle index_r, a \rangle\}$ follows $tSet' \subset tSet$.
From this and $MaxIndex(tSet) < index_s$ follows $tSet' = \emptyset \vee MaxIndex(tSet') < index_s$.
From this and UNCHANGED $index_s$ follows $tSet' = \emptyset \vee MaxIndex(tSet') < index_s'$. □

$\langle 9 \rangle 2.$ $\wedge\ |tSet| = index_s - index_r$
$\wedge\ \langle index_r, a \rangle \in tSet$
$\wedge\ tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index_r' = index_r + 1$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow |tSet'| = index_s' - index_r'$

PROOF: From $\langle index_r, a \rangle \in tSet$ and $tSet' = tSet \setminus \{\langle index_r, a \rangle\}$ follows $|tSet'| = |tSet| - 1$. From $index_r' = index_r + 1$ and UNCHANGED $index_s$ follows $index_s' - index_r' = index_s - index_r - 1$. Now we obtain the following equality:

$$|tSet'| = |tSet| - 1$$
$$= index_s - index_r - 1$$
$$= index'_s - index'_r. \square$$

$\langle 9 \rangle 3.$ $\land \forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$
$\land tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\land index'_r = index_r + 1$
$\land$ UNCHANGED $index_s$
$\Rightarrow \forall i \in 1 \,..\, (index'_s - index'_r) : |\{p \in tSet' : p[1] = i + index'_r - 1\}| = 1$

PROOF: From $\forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$ follows $\forall i \in 2 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$, and this is equivalent to $\forall i \in 1 \,..\, (index_s - index_r) - 1 : |\{p \in tSet : p[1] = i + index_r\}| = 1$. From this, $index'_r = index_r + 1$ and UNCHANGED $index_s$ follows $\forall i \in 1 \,..\, (index'_s - index'_r) : |\{p \in tSet : p[1] = i + index'_r - 1\}| = 1$. From this and $tSet' = tSet \setminus \{\langle index_r, a \rangle\}$ follows $\forall i \in 1 \,..\, (index'_s - index'_r) : |\{p \in tSet' : p[1] = i + index'_r - 1\}| = 1. \square$

$\langle 9 \rangle 4.$ Q.E.D.
PROOF: Propositional logic from $\langle 9 \rangle 1$, $\langle 9 \rangle 2$ and $\langle 9 \rangle 3$. $\square$

$\langle 8 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definition of *Pop*. $\square$

$\langle 6 \rangle 3.$ $\land$ *Inv*
$\land$ UNCHANGED *Vars*
$\Rightarrow Inv'$

PROOF: Direct from the definitions of *Vars* and *Inv*. $\square$

$\langle 6 \rangle 4.$ Q.E.D.
PROOF: Propositional logic from $\langle 6 \rangle 1$, $\langle 6 \rangle 2$ and $\langle 6 \rangle 3$. $\square$

$\langle 5 \rangle 2.$ Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. $\square$

$\langle 4 \rangle 2.$ Q.E.D.
PROOF: Direct from $\langle 4 \rangle 1$ using rule INV1. $\square$

$\langle 3 \rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$. $\square$

$\langle 2 \rangle 2.$ $\land \Box Inv$
$\land \Box[SomeAction]_{Vars}$
$\Rightarrow \Box[HLS.SomeAction]_{buffer}$

PROOF:
$\langle 3 \rangle 1.$ $\land$ *Inv*
$\land [SomeAction]_{Vars}$

$\Rightarrow [HLS.SomeAction]_{buffer}$

PROOF:

$\langle 4 \rangle 1.$ $\land\ Inv$
$\quad \land \lor \exists a : Push(a)$
$\quad\quad \lor\ Pop$
$\quad\quad \lor$ UNCHANGED $Vars$
$\quad \Rightarrow \lor \exists a : HLS.Push(a)$
$\quad\quad \lor\ HLS.Pop$
$\quad\quad \lor$ UNCHANGED $buffer$

PROOF:

$\langle 5 \rangle 1.$ $\land\ Inv \quad \Rightarrow \exists a : HLS.Push(a)$
$\quad \land \exists a : Push(a)$

PROOF:

LET: a : CONSTANT

$\langle 6 \rangle 1.$ $\land\ Inv \quad \Rightarrow HLS.Push(a)$
$\quad \land\ Push(a)$

PROOF:

$\langle 7 \rangle 1.$ $\land\ Inv$
$\quad \land\ a \in Messages$
$\quad \land\ index_s - index_r < BufferLength$
$\quad \land\ tSet' = tSet \cup \{\langle index_s, a \rangle\}$
$\quad \land\ index'_s = index_s + 1$
$\quad \land$ UNCHANGED $index_r$
$\quad \Rightarrow \land\ a \in Messages$
$\quad\quad \land\ Len(buffer) < BufferLength$
$\quad\quad \land\ buffer' = buffer \circ \langle a \rangle$

PROOF:

$\langle 8 \rangle 1.$ $a \in Messages \Rightarrow a \in Messages$

PROOF: Trivial. □

$\langle 8 \rangle 2.$ $index_s - index_r < BufferLength \Rightarrow Len(buffer) < BufferLength$

PROOF: From the definition of $buffer$ follows $Len(buffer) = index_s - index_r$. From this and $index_s - index_r < BufferLength$ follows $Len(buffer) < BufferLength$. □

$\langle 8 \rangle 3.$ $\land \lor\ tSet = \emptyset$
$\quad\quad \lor\ MaxIndex(tSet) < index_s$
$\quad \land\ \forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$
$\quad \land\ tSet' = tSet \cup \{\langle index_s, a \rangle\}$
$\quad \land\ index'_s = index_s + 1$
$\quad \land$ UNCHANGED $index_r$
$\quad \Rightarrow buffer' = buffer \circ \langle a \rangle$

PROOF: For all $i \in 1 \,..\, (index_s - index_r)$ holds $i + index_r - 1 < index_s$. From this and

$\forall i \in 1 \,..\, (index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$ follows
$[i \in 1 \,..\, (index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$= [i \in 1 \,..\, (index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet]$.
From $tSet = \emptyset \vee MaxIndex(tSet) < index_s$ follows
$\text{CHOOSE } x : \langle index_s, x \rangle \in tSet \cup \{\langle index_s, a \rangle\} = a$.
From the definition of $buffer$ follows $Len(buffer) = index_s - index_r$ and thus
$buffer \sqcup [index_s - index_r + 1 \mapsto a] = buffer \circ \langle a \rangle$.
Now we obtain the following equality chain:
$buffer'$
$= [i \in 1 \,..\, (index'_s - index'_r) \mapsto \text{CHOOSE } x : \langle i + index'_r - 1, x \rangle \in tSet']$
$= [i \in 1 \,..\, (index_s + 1 - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$= [i \in 1 \,..\, (index_s - index_r) \cup \{index_s - index_r + 1\}$
$\quad \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$= [i \in 1 \,..\, (index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$\quad \sqcup [index_s - index_r + 1 \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$= [i \in 1 \,..\, (index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$\quad \sqcup [index_s - index_r + 1 \mapsto \text{CHOOSE } x : \langle index_s, x \rangle \in tSet \cup \{\langle index_s, a \rangle\}]$
$= [i \in 1 \,..\, (index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x \rangle \in tSet]$
$\quad \sqcup [index_s - index_r + 1 \mapsto a]$
$= buffer \sqcup [index_s - index_r + 1 \mapsto a]$
$= buffer \circ \langle a \rangle$. □

$\langle 8 \rangle$4. Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$, $\langle 8 \rangle 2$ and $\langle 8 \rangle 3$. □

$\langle 7 \rangle$2. Q.E.D.
PROOF: Direct from the definitions of $Push(a)$ and $HLS.Push(a)$. □

$\langle 6 \rangle$2. Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

$\langle 5 \rangle$2. $\wedge\ Inv$
$\wedge\ Pop$
$\Rightarrow HLS.Pop$

PROOF:
$\langle 6 \rangle$1. $\wedge\ Inv$
$\wedge\ \exists\, a : \wedge\ \langle index_r, a \rangle \in tSet$
$\qquad \wedge\ tSet' = tSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index'_r = index_r + 1$
$\wedge\ \text{UNCHANGED } index_s$
$\Rightarrow \wedge\ buffer \neq \langle\, \rangle$
$\quad \wedge\ buffer' = Tail(buffer)$

PROOF:
LET: a : CONSTANT

$\langle 7\rangle 1.$ $\land$ Inv
$\land$ $\langle index_r, a\rangle \in tSet$
$\land$ $tSet' = tSet \setminus \{\langle index_r, a\rangle\}$
$\land$ $index'_r = index_r + 1$
$\land$ UNCHANGED $index_s$
$\Rightarrow \land$ $buffer \neq \langle\,\rangle$
$\land$ $buffer' = Tail(buffer)$

PROOF:
$\langle 8\rangle 1.$ $\land$ $|tSet| = index_s - index_r$
$\land$ $\langle index_r, a\rangle \in tSet$
$\Rightarrow$ $buffer \neq \langle\,\rangle$

PROOF: From $|tSet| = index_s - index_r$ and the definition of $buffer$ follows $Len(buffer) = |tSet|$. From this and $\langle index_r, a\rangle \in tSet$ follows $buffer \neq \langle\,\rangle$. □

$\langle 8\rangle 2.$ $\land \lor$ $tSet = \emptyset$
$\lor$ $MaxIndex(tSet) < index_s$
$\land$ $|tSet| = index_s - index_r$
$\land$ $\forall\, i \in 1\,..\,(index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$
$\land$ $\langle index_r, a\rangle \in tSet$
$\land$ $tSet' = tSet \setminus \{\langle index_r, a\rangle\}$
$\land$ $index'_r = index_r + 1$
$\land$ UNCHANGED $index_s$
$\Rightarrow$ $buffer' = Tail(buffer)$

PROOF: For all $i \in 2\,..\,(index_s - index_r)$ holds $i + index_r - 1 > index_r$. From this and $\forall\, i \in 1\,..\,(index_s - index_r) : |\{p \in tSet : p[1] = i + index_r - 1\}| = 1$ follows
$[i \in 2\,..\,(index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x\rangle \in tSet \setminus \{\langle index_r, a\rangle\}]$
$= [i \in 2\,..\,(index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x\rangle \in tSet]$.
Now we obtain the following equality chain:
$buffer'$
$= [i \in 1\,..\,(index'_s - index'_r) \mapsto \text{CHOOSE } x : \langle i + index'_r - 1, x\rangle \in tSet']$
$= [i \in 1\,..\,(index_s - index_r) - 1 \mapsto \text{CHOOSE } x : \langle i + index_r, x\rangle \in tSet \setminus \{\langle index_r, a\rangle\}]$
$= Tail([1 \mapsto \text{CHOOSE } x : \langle index_r, x\rangle \in tSet] \sqcup$
$[i \in 2\,..\,(index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x\rangle \in tSet \setminus \{\langle index_r, a\rangle\}])$
$= Tail([1 \mapsto \text{CHOOSE } x : \langle 1 + index_r - 1, x\rangle \in tSet] \sqcup$
$[i \in 2\,..\,(index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x\rangle \in tSet])$
$= Tail([i \in 1\,..\,(index_s - index_r) \mapsto \text{CHOOSE } x : \langle i + index_r - 1, x\rangle \in tSet])$
$= Tail(buffer)$. □

$\langle 8\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 8\rangle 1$ and $\langle 8\rangle 2$. □

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

$\langle 6\rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of Pop and $HLS.Pop$. □

$\langle 5\rangle 3.$ UNCHANGED $Vars$ $\Rightarrow$ UNCHANGED $buffer$

PROOF: Direct from the definitions of *Vars* and *buffer*. □

$\langle 5 \rangle 4$. Q.E.D.
PROOF: Propositional logic from $\langle 5 \rangle 1$, $\langle 5 \rangle 2$ and $\langle 5 \rangle 3$. □

$\langle 4 \rangle 2$. Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. □

$\langle 3 \rangle 2$. Q.E.D.
PROOF: Proof by STL5 and STL4. □

$\langle 2 \rangle 3$. Q.E.D.
PROOF: Direct from $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$ using propositional logic. □

$\langle 1 \rangle 3$. Q.E.D.
PROOF: Direct from $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ using propositional logic. □

4 Sender and Receiver

In this module, we split *tSet* into two different variables, *rSet* and *sSet*. We are slowly evolving towards a specification with distinct sender and receiver sides. At this stage though, we keep $index_r$ and $index_s$ as global variables. In particular, the sender action *Push* uses the information in the receiver variable $index_r$.

The buffer is now the (disjoint) union of *rSet* and *sSet*. Its actual size still equals the difference between $index_s$ and $index_r$.

The only changes to *Push* and *Pop* are that we now *Push* to *sSet* but *Pop* from *rSet*. A new action *Transfer* is added to the model, to represent the transfer of information from *rSet* to *sSet*. This action picks non-deterministically a pair from *sSet* and moves it to *rSet*. Indexing and $index_r$ become critical as pairs are *Transfer*'d in non-deterministic order, but must be *Pop*'d in FIFO order.

module *III*

$\textsc{extends}\ \mathit{CommonDefinitions}$

$\textsc{variables}\ \mathit{sSet}, \mathit{rSet}, \mathit{index}_s, \mathit{index}_r$

$$\mathit{Vars} \triangleq \langle \mathit{sSet}, \mathit{rSet}, \mathit{index}_s, \mathit{index}_r \rangle$$

$$\begin{aligned}\mathit{Init} \triangleq\ & \land\ \mathit{index}_r = 0\\ & \land\ \mathit{index}_s = 0\\ & \land\ \mathit{sSet} = \emptyset\\ & \land\ \mathit{rSet} = \emptyset\end{aligned}$$

$$\begin{aligned}\mathit{Push}(a) \triangleq\ & \land\ a \in \mathit{Messages}\\ & \land\ \mathit{index}_s - \mathit{index}_r < \mathit{BufferLength}\\ & \land\ \mathit{sSet}' = \mathit{sSet} \cup \{\langle \mathit{index}_s, a\rangle\}\\ & \land\ \mathit{index}'_s = \mathit{index}_s + 1\\ & \land\ \textsc{unchanged}\ \langle \mathit{rSet}, \mathit{index}_r \rangle\end{aligned}$$

$$\begin{aligned}\mathit{Pop} \triangleq\ & \land\ \exists\, a : \land\ \langle \mathit{index}_r, a\rangle \in \mathit{rSet}\\ & \qquad\quad \land\ \mathit{rSet}' = \mathit{rSet} \setminus \{\langle \mathit{index}_r, a\rangle\}\\ & \land\ \mathit{index}'_r = \mathit{index}_r + 1\\ & \land\ \textsc{unchanged}\ \langle \mathit{sSet}, \mathit{index}_s \rangle\end{aligned}$$

$$\begin{aligned}\mathit{Transfer} \triangleq\ & \land\ \exists\, p : \land\ p \in \mathit{sSet}\\ & \qquad\quad \land\ \mathit{sSet}' = \mathit{sSet} \setminus \{p\}\\ & \qquad\quad \land\ \mathit{rSet}' = \mathit{rSet} \cup \{p\}\\ & \land\ \textsc{unchanged}\ \langle \mathit{index}_s, \mathit{index}_r \rangle\end{aligned}$$

$$\begin{aligned}\mathit{SomeAction} \triangleq\ & \lor\ \exists\, a : \mathit{Push}(a)\\ & \lor\ \mathit{Pop}\\ & \lor\ \mathit{Transfer}\end{aligned}$$

$$\begin{aligned}\mathit{Safety} \triangleq\ & \land\ \mathit{Init}\\ & \land\ \Box[\mathit{SomeAction}]_{\mathit{Vars}}\end{aligned}$$

III.Safety implies II.Safety

module *III implies II*

EXTENDS *III*

$HLS \triangleq$ INSTANCE *II* WITH $tSet \leftarrow sSet \cup rSet$
$\quad index_s \leftarrow index_s$
$\quad index_r \leftarrow index_r$

THEOREM $Safety \Rightarrow HLS.Safety$

$\langle 0\rangle 2.$ $Safety \Rightarrow HLS.Safety$

PROOF:
$\langle 1\rangle 1.$ $Init \Rightarrow HLS.Init$

PROOF:
$\langle 2\rangle 1.$
$$\begin{array}{l} \land\ sSet = \emptyset \\ \land\ rSet = \emptyset \\ \land\ index_s = 0 \\ \land\ index_r = 0 \\ \Rightarrow \land\ sSet \cup rSet = \emptyset \\ \quad\ \land\ index_s = 0 \\ \quad\ \land\ index_r = 0 \end{array}$$

PROOF: Immediate. □

$\langle 2\rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of *Init* and *HLS.Init*. □

$\langle 1\rangle 2.$
$$\begin{array}{l} \land\ Init \\ \land\ \Box[SomeAction]_{Vars} \\ \Rightarrow \Box[HLS.SomeAction]_{HLS.Vars} \end{array}$$

PROOF: We use the following invariant:
LET: $Inv \triangleq$
$$\begin{array}{l} 1.\land\ sSet \cap rSet = \emptyset \\ 2.\land\ \lor\ sSet \cup rSet = \emptyset \\ \quad\ \ \lor\ MaxIndex(sSet \cup rSet) < index_s \end{array}$$

We use *Inv*.1 to prove that *Pop* simulates *HLS.Pop*.
We use *Inv*.2 to prove that *Inv*.1 is invariant under *Push*.

$\langle 2\rangle 1.$
$$\begin{array}{l} \land\ Init \\ \land\ \Box[SomeAction]_{Vars} \\ \Rightarrow \Box Inv \end{array}$$

PROOF:
$\langle 3\rangle 1.$ $Init \Rightarrow Inv$

PROOF: $Inv.1$ and $Inv.2$ follow immediately from $sSet = \emptyset$ and $rSet = \emptyset$. □

⟨3⟩2. $\land\ Inv$
$\land\ \Box[SomeAction]_{Vars}$
$\Rightarrow \Box Inv$

PROOF:
⟨4⟩1. $\land\ Inv$
$\land\ [SomeAction]_{Vars}$
$\Rightarrow Inv'$

PROOF:
⟨5⟩1. $\land\ Inv$
$\land\ \lor\ \exists\, a : Push(a)$
$\lor\ Pop$
$\lor\ Transfer$
$\lor$ UNCHANGED $Vars$
$\Rightarrow Inv'$

PROOF:
⟨6⟩1. $\land\ Inv$
$\land\ \exists\, a : Push(a)$
$\Rightarrow Inv'$

PROOF:
LET: a : CONSTANT

⟨7⟩1. $\land\ Inv$
$\land\ Push(a)$
$\Rightarrow Inv'$

PROOF:
⟨8⟩1. $\land\ Inv$
$\land\ a \in Messages$
$\land\ index_s - index_r < BufferLength$
$\land\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\land\ index_s' = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r\rangle$
$\Rightarrow Inv'$

PROOF:
⟨9⟩1. $\land\ sSet \cap rSet = \emptyset$
$\land\ \lor\ sSet \cup rSet = \emptyset$
$\lor\ MaxIndex(sSet \cup rSet) < index_s$
$\land\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\land$ UNCHANGED $rSet$
$\Rightarrow sSet' \cap rSet' = \emptyset$

PROOF: From $sSet \cup rSet = \emptyset \lor MaxIndex(sSet \cup rSet) < index_s$ follows $\{\langle index_s, a\rangle\} \notin rSet$.
From this, the other parts of the premise and set theory we obtain

$$
\begin{aligned}
sSet' \cap rSet' &= (sSet \cup \{\langle index_s, a\rangle\}) \cap rSet \\
&= (sSet \cap rSet) \cup (\{\langle index_s, a\rangle\} \cap rSet) \\
&= \emptyset \cup \emptyset \\
&= \emptyset. \square
\end{aligned}
$$

$\langle 9\rangle 2.$
$$
\begin{array}{l}
\land \lor\ sSet \cup rSet = \emptyset \\
\quad \lor\ MaxIndex(sSet \cup rSet) < index_s \\
\land\ sSet' = sSet \cup \{\langle index_s, a\rangle\} \\
\land\ index'_s = index_s + 1 \\
\land\ \text{UNCHANGED } rSet \\
\Rightarrow \lor\ sSet' \cup rSet' = \emptyset \\
\quad\ \lor\ MaxIndex(sSet' \cup rSet') < index'_s
\end{array}
$$

PROOF: From $sSet \cup rSet = \emptyset \lor MaxIndex(sSet \cup rSet) < index_s$, $sSet' = sSet \cup \{\langle index_s, a\rangle\}$ and UNCHANGED $rSet$ follows $MaxIndex(sSet' \cup rSet') = index_s$.
From this and $index'_s = index_s + 1$ follows $MaxIndex(sSet' \cup rSet') < index'_s$. $\square$

$\langle 9\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$ and $\langle 9\rangle 2$. $\square$

$\langle 8\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of $Push(a)$. $\square$

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 6\rangle 2.$
$$
\begin{array}{l}
\land\ Inv \\
\land\ Pop \\
\Rightarrow Inv'
\end{array}
$$

PROOF:

$\langle 7\rangle 1.$
$$
\begin{array}{l}
\land\ Inv \\
\land\ \exists\, a : \land\ \langle index_r, a\rangle \in rSet \\
\qquad\quad\ \land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\} \\
\land\ index'_r = index_r + 1 \\
\land\ \text{UNCHANGED } \langle sSet, index_s\rangle \\
\Rightarrow Inv'
\end{array}
$$

PROOF:
LET: a : CONSTANT

$\langle 8\rangle 1.$
$$
\begin{array}{l}
\land\ Inv \\
\land\ \langle index_r, a\rangle \in rSet \\
\land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\} \\
\land\ index'_r = index_r + 1 \\
\land\ \text{UNCHANGED } \langle sSet, index_s\rangle \\
\Rightarrow Inv'
\end{array}
$$

PROOF:

$\langle 9\rangle 1.$ $\land\ sSet \cap rSet = \emptyset$
$\quad\land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}$
$\quad\land\ \text{UNCHANGED } sSet$
$\quad\Rightarrow sSet' \cap rSet' = \emptyset$

PROOF: From UNCHANGED $sSet$ and $rSet' = rSet \setminus \{\langle index_r, a\rangle\}$ follows $sSet' \cap rSet' \subseteq sSet \cap rSet$.
From this and $sSet \cap rSet = \emptyset$ follows $sSet' \cap rSet' = \emptyset$. □

$\langle 9\rangle 2.$ $\land\ \lor\ sSet \cup rSet = \emptyset$
$\quad\quad\lor\ MaxIndex(sSet \cup rSet) < index_s$
$\quad\land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}$
$\quad\land\ \text{UNCHANGED } \langle sSet, index_s\rangle$
$\quad\Rightarrow \lor\ sSet' \cup rSet' = \emptyset$
$\quad\quad\lor\ MaxIndex(sSet' \cup rSet') < index'_s$

PROOF: From UNCHANGED $sSet$ and $rSet' = rSet \setminus \{\langle index_r, a\rangle\}$ follows $sSet' \cap rSet' \subseteq sSet \cup rSet$.
From this, $sSet \cup rSet = \emptyset \lor MaxIndex(sSet \cup rSet) < index_s$ and UNCHANGED $index_s$ follows $sSet' \cup rSet' = \emptyset \lor MaxIndex(sSet' \cup rSet') < index'_s$. □

$\langle 9\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$ and $\langle 9\rangle 2$. □

$\langle 8\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. □

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of Pop. □

$\langle 6\rangle 3.$ $\land\ Inv$
$\quad\land\ Transfer$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 7\rangle 1.$ $\land\ Inv$
$\quad\land\ \exists p : \land\ p \in sSet$
$\quad\quad\quad\land\ sSet' = sSet \setminus \{p\}$
$\quad\quad\quad\land\ rSet' = rSet \cup \{p\}$
$\quad\land\ \text{UNCHANGED } \langle index_s, index_r\rangle$
$\quad\Rightarrow Inv'$

PROOF:
LET: $p \in sSet$

$\langle 8\rangle 1.$ $\land\ Inv$
$\quad\land\ sSet' = sSet \setminus \{p\}$
$\quad\land\ rSet' = rSet \cup \{p\}$
$\quad\land\ \text{UNCHANGED } \langle index_s, index_r\rangle$
$\quad\Rightarrow Inv'$

PROOF:

$\langle 9\rangle 1.$ $\wedge\ sSet \cap rSet = \emptyset$
$\wedge\ sSet' = sSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$
$\Rightarrow sSet' \cap rSet' = \emptyset$

PROOF: Set theory:

$$\begin{aligned} sSet' \cap rSet' &= (sSet \setminus \{p\}) \cap (rSet \cup \{p\}) \\ &= ((sSet \setminus \{p\}) \cap rSet) \cup ((sSet \setminus \{p\}) \cap \{p\}) \\ &= ((sSet \cap rSet) \setminus \{p\}) \cup \emptyset \\ &\subseteq sSet \cap rSet \\ &= \emptyset. \ \square \end{aligned}$$

$\langle 9\rangle 2.$ $\wedge \vee\ sSet \cup rSet = \emptyset$
$\vee\ MaxIndex(sSet \cup rSet) < index_s$
$\wedge\ sSet' = sSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow \vee\ sSet' \cup rSet' = \emptyset$
$\vee\ MaxIndex(sSet' \cup rSet') < index'_s$

PROOF: From $sSet' = sSet \setminus \{p\}$ and $rSet' = rSet \cup \{p\}$ follows $sSet' \cup rSet' = sSet \cup rSet \cup \{p\}$.
From this and $p \in sSet$ follows $sSet' \cup rSet' = sSet \cup rSet$.
From this, UNCHANGED $index_s$ and $sSet \cup rSet = \emptyset \vee MaxIndex(sSet \cup rSet) < index_s$ follows $sSet' \cup rSet' = \emptyset \vee MaxIndex(sSet' \cup rSet') < index'_s$. $\square$

$\langle 9\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$ and $\langle 9\rangle 2$. $\square$

$\langle 8\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of *Transfer*. $\square$

$\langle 6\rangle 4.$ $\wedge\ Inv$
$\wedge$ UNCHANGED $Vars$
$\Rightarrow Inv'$

PROOF: Direct from the definitions of *Vars* and *Inv*. $\square$

$\langle 6\rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 6\rangle 1$, $\langle 6\rangle 2$, $\langle 6\rangle 3$ and $\langle 6\rangle 4$. $\square$

$\langle 5\rangle 2.$ Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. $\square$

$\langle 4\rangle 2.$ Q.E.D.
PROOF: Direct from $\langle 4\rangle 1$ using rule INV1. $\square$

$\langle 3\rangle 3.$ Q.E.D.

PROOF: Propositional logic from $\langle 3\rangle 1$ and $\langle 3\rangle 2$. □

$\langle 2\rangle 2.$ $\wedge \Box Inv$
$\wedge \Box[SomeAction]_{Vars}$
$\Rightarrow \Box[HLS.SomeAction]_{HLS.Vars}$

PROOF:

$\langle 3\rangle 1.$ $\wedge\ Inv$
$\wedge\ [SomeAction]_{Vars}$
$\Rightarrow [HLS.SomeAction]_{HLS.Vars}$

PROOF:

$\langle 4\rangle 1.$ $\wedge\ Inv$
$\wedge \vee \exists a : Push(a)$
$\quad \vee\ Pop$
$\quad \vee\ Transfer$
$\quad \vee$ UNCHANGED $Vars$
$\Rightarrow \vee \exists a : HLS.Push(a)$
$\quad \vee\ HLS.Pop$
$\quad \vee$ UNCHANGED $HLS.Vars$

PROOF:

$\langle 5\rangle 1.$ $\exists a : Push(a) \Rightarrow \exists a : HLS.Push(a)$

PROOF:

LET: a : CONSTANT

$\langle 6\rangle 1.$ $Push(a) \Rightarrow HLS.Push(a)$

PROOF:

$\langle 7\rangle 1.$ $\wedge\ a \in Messages$
$\wedge\ index_s - index_r < BufferLength$
$\wedge\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\wedge\ index_s' = index_s + 1$
$\wedge$ UNCHANGED $\langle rSet, index_r\rangle$
$\Rightarrow \wedge\ a \in Messages$
$\quad \wedge\ index_s - index_r < BufferLength$
$\quad \wedge\ sSet' \cup rSet' = sSet \cup rSet \cup \{\langle index_s, a\rangle\}$
$\quad \wedge\ index_s' = index_s + 1$
$\quad \wedge$ UNCHANGED $index_r$

PROOF:

$\langle 8\rangle 1.$ $a \in Messages \Rightarrow a \in Messages$

PROOF: Trivial. □

$\langle 8\rangle 2.$ $index_s - index_r < BufferLength \Rightarrow index_s - index_r < BufferLength$

PROOF: Trivial. □

$\langle 8\rangle 3.$ $\land\ sSet' = sSet \cup \{\langle\, index_s, a\,\rangle\}$
$\land$ UNCHANGED $rSet$
$\Rightarrow sSet' \cup rSet' = sSet \cup rSet \cup \{\langle\, index_s, a\,\rangle\}$

PROOF: Simple set theory. □

$\langle 8\rangle 4.$ $index'_s = index_s + 1 \Rightarrow index'_s = index_s + 1$

PROOF: Trivial. □

$\langle 8\rangle 5.$ UNCHANGED $index_r \Rightarrow$ UNCHANGED $index_r$

PROOF: Trivial. □

$\langle 8\rangle 6.$ Q.E.D.
PROOF: Propositional logic from $\langle 8\rangle 1$, $\langle 8\rangle 2$, $\langle 8\rangle 3$, $\langle 8\rangle 4$ and $\langle 8\rangle 5$. □

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of $Push(a)$ and $HLS.Push(a)$. □

$\langle 6\rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

$\langle 5\rangle 2.$ $\land\ Inv$
$\land\ Pop$
$\Rightarrow HLS.Pop$

PROOF:
$\langle 6\rangle 1.$ $\land\ Inv$
$\land\ \exists\, a : \land\ \langle\, index_r, a\,\rangle \in rSet$
$\qquad \land\ rSet' = rSet \setminus \{\langle\, index_r, a\,\rangle\}$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $\langle\, sSet, index_s\,\rangle$
$\Rightarrow \land\ \exists\, a : \land\ \langle\, index_r, a\,\rangle \in sSet \cup rSet$
$\qquad \land\ sSet' \cup rSet' = (sSet \cup rSet) \setminus \{\langle\, index_r, a\,\rangle\}$
$\quad \land\ index'_r = index_r + 1$
$\quad \land$ UNCHANGED $index_s$

PROOF:
LET: a : CONSTANT

$\langle 7\rangle 1.$ $\land\ Inv$
$\land\ \langle\, index_r, a\,\rangle \in rSet$
$\land\ rSet' = rSet \setminus \{\langle\, index_r, a\,\rangle\}$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $\langle\, sSet, index_s\,\rangle$
$\Rightarrow \land\ \langle\, index_r, a\,\rangle \in sSet \cup rSet$
$\quad \land\ sSet' \cup rSet' = (sSet \cup rSet) \setminus \{\langle\, index_r, a\,\rangle\}$
$\quad \land\ index'_r = index_r + 1$
$\quad \land$ UNCHANGED $index_s$

PROOF:

$\langle 8 \rangle 1.$ $\langle index_r, a \rangle \in rSet \Rightarrow \langle index_r, a \rangle \in sSet \cup rSet$

PROOF: Simple set theory. □

$\langle 8 \rangle 2.$ $\wedge\ sSet \cap rSet = \emptyset$
$\wedge\ \langle index_r, a \rangle \in rSet$
$\wedge\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$ UNCHANGED $sSet$
$\Rightarrow sSet' \cup rSet' = (sSet \cup rSet) \setminus \{\langle index_r, a \rangle\}$

PROOF: From $sSet \cap rSet = \emptyset$ and $\langle index_r, a \rangle \in rSet$ follows $\langle index_r, a \rangle \notin sSet$ and thus $sSet = sSet \setminus \{\langle index_r, a \rangle\}$.
Using this, $rSet' = rSet \setminus \{\langle index_r, a \rangle\}$, UNCHANGED $sSet$ and some set theory we obtain

$$
\begin{aligned}
sSet' \cup rSet' &= sSet \cup (rSet \setminus \{\langle index_r, a \rangle\}) \\
&= (sSet \setminus \{\langle index_r, a \rangle\}) \cup (rSet \setminus \{\langle index_r, a \rangle\}) \\
&= (sSet \cup rSet) \setminus \{\langle index_r, a \rangle\}.
\end{aligned}
$$
□

$\langle 8 \rangle 3.$ $index'_r = index_r + 1 \Rightarrow index'_r = index_r + 1$

PROOF: Trivial. □

$\langle 8 \rangle 4.$ UNCHANGED $index_s \Rightarrow$ UNCHANGED $index_s$

PROOF: Trivial. □

$\langle 8 \rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$, $\langle 8 \rangle 2$, $\langle 8 \rangle 3$ and $\langle 8 \rangle 4$. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

$\langle 6 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of Pop and $HLS.Pop$. □

$\langle 5 \rangle 3.$ $Transfer \Rightarrow$ UNCHANGED $HLS.Vars$

PROOF:
$\langle 6 \rangle 1.$ $\wedge\ \exists\, p : \wedge\ p \in sSet$
$\wedge\ sSet' = sSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle index_s, index_r \rangle$
$\Rightarrow$ UNCHANGED $HLS.Vars$

PROOF:
LET: $p \in sSet$

$\langle 7 \rangle 1.$ $\wedge\ sSet' = sSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle index_s, index_r \rangle$
$\Rightarrow$ UNCHANGED $HLS.Vars$

PROOF: From $sSet' = sSet \setminus \{p\}$ and $rSet' = rSet \cup \{p\}$ follows $sSet' \cup rSet' = sSet \cup$

$rSet \cup \{p\}$.
From this and $p \in sSet$ follows $sSet' \cup rSet' = sSet \cup rSet$.
From this, UNCHANGED $\langle index_s, index_r \rangle$ and the definition of $HLS.Vars$ follows UNCHANGED $HLS.Vars$. □

$\langle 7 \rangle 2$. Q.E.D.
PROOF: Predicate logic: forall-intro. □

$\langle 6 \rangle 2$. Q.E.D.
PROOF: Direct from the definition of $Transfer$. □

$\langle 5 \rangle 4$. UNCHANGED $Vars \Rightarrow$ UNCHANGED $HLS.Vars$

PROOF: Direct from the definitions of $Vars$ and $HLS.Vars$. □

$\langle 5 \rangle 5$. Q.E.D.
PROOF: Propositional logic from $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, $\langle 5 \rangle 3$ and $\langle 5 \rangle 4$. □

$\langle 4 \rangle 2$. Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. □

$\langle 3 \rangle 2$. Q.E.D.
PROOF: Proof by STL5 and STL4. □

$\langle 2 \rangle 3$. Q.E.D.
PROOF: Direct from $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$ using propositional logic. □

$\langle 1 \rangle 3$. Q.E.D.
PROOF: Direct from $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ using propositional logic. □

5 Copy and Purge

Specification *IV* introduces a first degree of asynchronism between both sides of communication. An index-message pair is no longer *Transfer*'d but rather *Copy*'d from *sSet* to *rSet* and later *Purge*'d from *sSet*, when it has been *Pop*'d for certain. As a consequence *rSet* is not disjoint from, but rather a subset of *sSet*. This asynchronism is a first step towards handling loss of data between sender and receiver: the sender became careful, she keeps a sent message until she knows it has been *Pop*'d.

In the Specifications *II* and *III*, the difference $index_s - index_r$ was controlled to ensure that the buffer didn't exceed the maximum length. Here, we use the term $index_s - MinIndex(sSet)$ instead, since $MinIndex(sSet)$ can linger below $index_r$. The size control is now completely local to the sender side.

This is the only change to *Push*. *Pop* has not changed at all. In contrast to *III.Transfer*, *Copy* allows multiple copying of the same pair to *rSet*. Since we are dealing with sets, such multiple copies are idempotent. *Purge* can handle several messages at once: it non-deterministically picks an already-*Pop*'d message in *sSet* and deletes this and all older messages.

The sender and the receiver side are not yet completely separated. *Copy* is a common action using both *sSet* and *rSet*. *Purge* could be regarded as a sender action, but it still uses the receiver information in $index_r$, thus $index_r$ remains a global variable.

module *IV*

EXTENDS *CommonDefinitions*

VARIABLES $sSet, rSet, index_r, index_s$

$$
\begin{aligned}
&Vars \triangleq \langle sSet, rSet, index_r, index_s \rangle\\
&Init \triangleq \land\ index_r = 0\\
&\qquad\quad \land\ index_s = 0\\
&\qquad\quad \land\ sSet = \emptyset\\
&\qquad\quad \land\ rSet = \emptyset
\end{aligned}
$$

$$
\begin{aligned}
&Push(a) \triangleq \land\ a \in Messages\\
&\qquad\qquad\quad \land \lor\ sSet = \emptyset\\
&\qquad\qquad\qquad \lor\ index_s - MinIndex(sSet) < BufferLength\\
&\qquad\qquad\quad \land\ sSet' = sSet \cup \{\langle index_s, a\rangle\}\\
&\qquad\qquad\quad \land\ index'_s = index_s + 1\\
&\qquad\qquad\quad \land\ \text{UNCHANGED } \langle rSet, index_r \rangle\\
&Pop \triangleq \land\ \exists\, a : \land\ \langle index_r, a\rangle \in rSet\\
&\qquad\qquad\qquad\quad \land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}\\
&\qquad\quad \land\ index'_r = index_r + 1\\
&\qquad\quad \land\ \text{UNCHANGED } \langle sSet, index_s \rangle\\
&Copy \triangleq \land\ \exists\, p : \land\ p \in sSet\\
&\qquad\qquad\qquad\quad \land\ p[1] \geq index_r\\
&\qquad\qquad\qquad\quad \land\ rSet' = rSet \cup \{p\}\\
&\qquad\quad \land\ \text{UNCHANGED } \langle sSet, index_s, index_r \rangle\\
&Purge \triangleq \land\ \exists\, p \in sSet : \land\ p[1] < index_r\\
&\qquad\qquad\qquad\qquad\quad \land\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}\\
&\qquad\quad \land\ \text{UNCHANGED } \langle rSet, index_r, index_s \rangle
\end{aligned}
$$

$$
\begin{aligned}
SomeAction \;&\triangleq\; \lor\; \exists\, a : Push(a)\\
&\;\; \lor\; Pop\\
&\;\; \lor\; Copy\\
&\;\; \lor\; Purge
\end{aligned}
$$

$$
\begin{aligned}
Safety \;&\triangleq\; \land\; Init\\
&\;\; \land\; \Box[SomeAction]_{Vars}
\end{aligned}
$$

IV.Safety implies III.Safety

module *IV implies III*

EXTENDS IV

$$UpperSet(set, index) \triangleq \{p \in set : p[1] \geq index\}$$
$$HLSsSet \triangleq UpperSet(sSet, index_r) \setminus rSet$$
$$HLS \triangleq \text{INSTANCE } III \text{ WITH } \begin{array}{l} sSet \leftarrow HLSsSet \\ rSet \leftarrow rSet \\ index_s \leftarrow index_s \\ index_r \leftarrow index_r \end{array}$$
$$Indices \triangleq \{i \in Nat : index_r \leq i < index_s\}$$

THEOREM $Safety \Rightarrow HLS.Safety$

$\langle 0 \rangle 3.$ $Safety \Rightarrow HLS.Safety$

PROOF:
$\langle 1 \rangle 1.$ $Init \Rightarrow HLS.Init$

PROOF:
$\langle 2 \rangle 1.$
$$\begin{array}{l} \land\ sSet = \emptyset \\ \land\ rSet = \emptyset \\ \land\ index_s = 0 \\ \land\ index_r = 0 \\ \Rightarrow \land\ HLSsSet = \emptyset \\ \quad \land\ rSet = \emptyset \\ \quad \land\ index_s = 0 \\ \quad \land\ index_r = 0 \end{array}$$

PROOF: Direct from the definition of $HLSsSet$. □

$\langle 2 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of $Init$ and $HLS.Init$. □

$\langle 1 \rangle 2.$
$$\begin{array}{l} \land\ Init \\ \land\ \Box[SomeAction]_{Vars} \\ \Rightarrow \Box[HLS.SomeAction]_{HLS.Vars} \end{array}$$

PROOF: We use the following invariant:
LET: $Inv \triangleq$
$$\begin{array}{l} 1.\land\ rSet \subseteq UpperSet(sSet, index_r) \\ 2.\land\ \lor\ sSet = \emptyset \\ \quad\ \ \lor\ MaxIndex(sSet) < index_s \\ 3.\land\ index_r \leq index_s \\ 4.\land\ \forall i \in Indices : |\{p \in sSet : p[1] = i\}| = 1 \end{array}$$

We use $Inv.1$, $Inv.2$, $Inv.3$ and $Inv.4$ to prove that $Push$ simulates $HLS.Push$.
We use $Inv.1$ and $Inv.4$ to prove that Pop simulates $HLS.Pop$.

Instead of $Inv.1$ one might use the weaker $rSet \subseteq sSet$, but that can't be proven to be invariant without additional Inv-parts.

$\langle 2 \rangle 1.$ $\land$ $Init$
$\land$ $\Box[SomeAction]_{Vars}$
$\Rightarrow \Box Inv$

PROOF:
$\langle 3 \rangle 1.$ $Init \Rightarrow Inv$

PROOF: $Inv.1$ follows from $sSet = \emptyset$ and $rSet = \emptyset$.
$Inv.2$ follows from $sSet = \emptyset$.
$Inv.3$ follows from $index_s = 0$ and $index_r = 0$.
$Inv.4$ follows from $Indices = \emptyset$, which follows from $index_s = 0$ and $index_r = 0$. □

$\langle 3 \rangle 2.$ $\land$ Inv
$\land$ $\Box[SomeAction]_{Vars}$
$\Rightarrow \Box Inv$

PROOF:
$\langle 4 \rangle 1.$ $\land$ Inv
$\land$ $[SomeAction]_{Vars}$
$\Rightarrow Inv'$

PROOF:
$\langle 5 \rangle 1.$ $\land$ Inv
$\land$ $\lor$ $\exists\, a : Push(a)$
$\lor$ Pop
$\lor$ $Copy$
$\lor$ $Purge$
$\lor$ UNCHANGED $Vars$
$\Rightarrow Inv'$

PROOF:
$\langle 6 \rangle 1.$ $\land$ Inv
$\land$ $\exists\, a : Push(a)$
$\Rightarrow Inv'$

PROOF:
LET: a : CONSTANT

$\langle 7 \rangle 1.$ $\land$ Inv
$\land$ $Push(a)$
$\Rightarrow Inv'$

PROOF:

$\langle 8 \rangle 1.$ $\land\ Inv$
$\land\ a \in Messages$
$\land\ \lor\ sSet = \emptyset$
$\quad \lor\ index_s - MinIndex(sSet) < BufferLength$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land\ index'_s = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$
$\Rightarrow Inv'$

PROOF:

$\langle 9 \rangle 1.$ $\land\ rSet \subseteq UpperSet(sSet, index_r)$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$
$\Rightarrow rSet' \subseteq UpperSet(sSet', index'_r)$

PROOF: Follows from
$rSet' = rSet \subseteq UpperSet(sSet, index_r) \subseteq UpperSet(sSet', index'_r)$. □

$\langle 9 \rangle 2.$ $\land\ \lor\ sSet = \emptyset$
$\quad \lor\ MaxIndex(sSet) < index_s$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land\ index'_s = index_s + 1$
$\Rightarrow \lor\ sSet' = \emptyset$
$\quad \lor\ MaxIndex(sSet') < index'_s$

PROOF: From $sSet = \emptyset \lor MaxIndex(sSet) < index_s$ and $sSet' = sSet \cup \{\langle index_s, a \rangle\}$ follows $MaxIndex(sSet') = index_s$.
From this and $index'_s = index_s + 1$ follows $MaxIndex(sSet') < index'_s$. □

$\langle 9 \rangle 3.$ $\land\ index_r \leq index_s$
$\land\ index'_s = index_s + 1$
$\land$ UNCHANGED $index_r$
$\Rightarrow index'_r \leq index'_s$

PROOF: Follows from $index'_r = index_r \leq index_s < index'_s$. □

$\langle 9 \rangle 4.$ $\land\ \lor\ sSet = \emptyset$
$\quad \lor\ MaxIndex(sSet) < index_s$
$\land\ \forall i \in Indices : |\{p \in sSet : p[1] = i\}| = 1$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land\ index'_s = index_s + 1$
$\land$ UNCHANGED $index_r$
$\Rightarrow \forall i \in Indices' : |\{p \in sSet' : p[1] = i\}| = 1$

PROOF: From $sSet' = sSet \cup \{\langle index_s, a \rangle\}$ follows for all $i \neq index_s$:
$|\{p \in sSet : p[1] = i\}| = |\{p \in sSet' : p[1] = i\}|$.
Thus for all $i \in Indices$: $|\{p \in sSet' : p[1] = i\}| = |\{p \in sSet : p[1] = i\}| = 1$.
From $sSet = \emptyset \lor MaxIndex(sSet) < index_s$ follows $|\{p \in sSet : p[1] = index_s\}| = 0$.
From this and $sSet' = sSet \cup \{\langle index_s, a \rangle\}$ follows
$|\{p \in sSet' : p[1] = index_s\}| = 1$.
From $index'_s = index_s + 1$, UNCHANGED $index_r$ and the definition of *Indices*

follows $Indices' = Indices \cup \{index_s\}$.
Now from this, $\forall\, i \in Indices \,:\, |\{p \in sSet' \,:\, p[1] = i\}| = 1$
and $|\{p \in sSet' \,:\, p[1] = index_s\}| = 1$ we obtain
$\forall\, i \in Indices' \,:\, |\{p \in sSet' \,:\, p[1] = i\}| = 1$. □

$\langle 9\rangle 5$. Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$, $\langle 9\rangle 2$, $\langle 9\rangle 3$ and $\langle 9\rangle 4$. □

$\langle 8\rangle 2$. Q.E.D.
PROOF: Direct from the definition of $Push(a)$. □

$\langle 7\rangle 2$. Q.E.D.
PROOF: Predicate logic: forall-intro. □

$\langle 6\rangle 2$. $\land\ Inv$
$\land\ Pop$
$\Rightarrow Inv'$

PROOF:
$\langle 7\rangle 1$. $\land\ Inv$
$\land\ \exists\, a \,:\, \land\ \langle index_r, a\rangle \in rSet$
$\qquad \land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}$
$\land\ index_r' = index_r + 1$
$\land$ UNCHANGED $\langle sSet, index_s\rangle$
$\Rightarrow Inv'$

PROOF:
LET: a : CONSTANT

$\langle 8\rangle 1$. $\land\ Inv$
$\land\ \langle index_r, a\rangle \in rSet$
$\land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}$
$\land\ index_r' = index_r + 1$
$\land$ UNCHANGED $\langle sSet, index_s\rangle$
$\Rightarrow Inv'$

PROOF:
$\langle 9\rangle 1$. $\land\ rSet \subseteq UpperSet(sSet, index_r)$
$\land\ \forall\, i \in Indices \,:\, |\{p \in sSet \,:\, p[1] = i\}| = 1$
$\land\ \langle index_r, a\rangle \in rSet$
$\land\ rSet' = rSet \setminus \{\langle index_r, a\rangle\}$
$\land\ index_r' = index_r + 1$
$\land$ UNCHANGED $sSet$
$\Rightarrow rSet' \subseteq UpperSet(sSet', index_r')$

PROOF: From $rSet \subseteq UpperSet(sSet, index_r)$
follows $rSet = UpperSet(rSet, index_r)$.
From $rSet \subseteq UpperSet(sSet, index_r)$ and $\forall\, i \in Indices \,:\, |\{p \in sSet \,:\, p[1] = i\}| = 1$
follows $|\{p \in rSet \,:\, p[1] = index_r\}| = 1$.
From $|\{p \in rSet \,:\, p[1] = index_r\}| = 1$ and $\langle index_r, a\rangle \in rSet$
follows $\{p \in rSet \,:\, p[1] = index_r\} = \langle index_r, a\rangle$.

Thus $rSet \setminus \{\langle index_r, a\rangle\} = UpperSet(rSet, index_r + 1)$. Now we obtain

$$\begin{aligned} rSet' &= rSet \setminus \{\langle index_r, a\rangle\} \\ &= UpperSet(rSet, index_r + 1) \\ &\subseteq UpperSet(sSet, index_r + 1) \\ &= UpperSet(sSet', index'_r). \ \square \end{aligned}$$

$\langle 9\rangle 2.$ $\wedge \vee\ sSet = \emptyset$
$\quad\quad \vee\ MaxIndex(sSet) < index_s$
$\quad \wedge$ UNCHANGED $\langle sSet, index_s\rangle$
$\quad \Rightarrow \vee\ sSet' = \emptyset$
$\quad\quad \vee\ MaxIndex(sSet') < index'_s$

PROOF: Trivial. $\square$

$\langle 9\rangle 3.$ $\wedge\ rSet \subseteq UpperSet(sSet, index_r)$
$\quad \wedge \vee\ sSet = \emptyset$
$\quad\quad \vee\ MaxIndex(sSet) < index_s$
$\quad \wedge\ index_r \leq index_s$
$\quad \wedge\ \langle index_r, a\rangle \in rSet$
$\quad \wedge\ index'_r = index_r + 1$
$\quad \wedge$ UNCHANGED $index_s$
$\quad \Rightarrow index'_r \leq index'_s$

PROOF: From $\langle index_r, a\rangle \in rSet$ and $rSet \subseteq UpperSet(sSet, index_r)$ follows $index_r \in sSet$.
From $index_r \in sSet$ and $sSet = \emptyset \vee MaxIndex(sSet) < index_s$ follows $index_r < index_s$ and thus $index_r + 1 \leq index_s$. Now we obtain $index'_r = index_r + 1 \leq index_s = index'_s$. $\square$

$\langle 9\rangle 4.$ $\wedge\ \forall\, i \in Indices\ :\ |\{p \in sSet\ :\ p[1] = i\}| = 1$
$\quad \wedge\ index'_r = index_r + 1$
$\quad \wedge$ UNCHANGED $\langle sSet, index_s\rangle$
$\quad \Rightarrow \forall\, i \in Indices'\ :\ |\{p \in sSet'\ :\ p[1] = i\}| = 1$

PROOF: From $index'_r = index_r + 1$ and UNCHANGED $index_s$ follows $Indices' \subseteq Indices$.
$\forall\, i \in Indices'\ :\ |\{p \in sSet'\ :\ p[1] = i\}| = 1$ follows from $Indices' \subseteq Indices$ and UNCHANGED $sSet$. $\square$

$\langle 9\rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$, $\langle 9\rangle 2$, $\langle 9\rangle 3$ and $\langle 9\rangle 4$. $\square$

$\langle 8\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of Pop. $\square$

$\langle 6\rangle 3.$ $\wedge\ Inv$
$\quad \wedge\ Copy$
$\quad \Rightarrow Inv'$

PROOF:
$\langle 7\rangle 1.$ $\land$ *Inv*
$\land \exists p : \land p \in sSet$
$\land p[1] \geq index_r$
$\land rSet' = rSet \cup \{p\}$
$\land$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow Inv'$

PROOF:
LET: $p \in sSet$

$\langle 8\rangle 1.$ $\land$ *Inv*
$\land p[1] \geq index_r$
$\land rSet' = rSet \cup \{p\}$
$\land$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow Inv'$

PROOF:
$\langle 9\rangle 1.$ $\land rSet \subseteq UpperSet(sSet, index_r)$
$\land p[1] \geq index_r$
$\land rSet' = rSet \cup \{p\}$
$\land$ UNCHANGED $\langle sSet, index_r \rangle$
$\Rightarrow rSet' \subseteq UpperSet(sSet', index'_r)$

PROOF: From $p \in sSet$ and $p[1] \geq index_r$ follows $p \in UpperSet(sSet, index_r)$. Thus $rSet' = rSet \cup \{p\} \subseteq UpperSet(sSet, index_r) = UpperSet(sSet', index'_r)$. □

$\langle 9\rangle 2.$ $\land \lor sSet = \emptyset$
$\lor MaxIndex(sSet) < index_s$
$\land$ UNCHANGED $\langle sSet, index_s \rangle$
$\Rightarrow \lor sSet' = \emptyset$
$\lor MaxIndex(sSet') < index'_s$

PROOF: Trivial. □

$\langle 9\rangle 3.$ $\land index_r \leq index_s$
$\land$ UNCHANGED $\langle index_s, index_r \rangle$
$\Rightarrow index'_r \leq index'_s$

PROOF: Trivial. □

$\langle 9\rangle 4.$ $\land \forall i \in Indices : |\{p \in sSet : p[1] = i\}| = 1$
$\land$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow \forall i \in Indices' : |\{p \in sSet' : p[1] = i\}| = 1$

PROOF: Direct from the definition of *Indices*. □

$\langle 9\rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$, $\langle 9\rangle 2$, $\langle 9\rangle 3$ and $\langle 9\rangle 4$. □

$\langle 8\rangle 2.$ Q.E.D.

PROOF: Predicate logic: forall-intro. $\square$

$\langle 7\rangle 2$. Q.E.D.
PROOF: Direct from the definition of *Copy*. $\square$

$\langle 6\rangle 4$. $\wedge$ *Inv*
$\wedge$ *Purge*
$\Rightarrow$ *Inv'*

PROOF:
$\langle 7\rangle 1$. $\wedge\ Inv$
$\wedge\ \exists\, p : \wedge\ p \in sSet$
$\qquad \wedge\ p[1] < index_r$
$\qquad \wedge\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\wedge$ UNCHANGED $\langle rSet, index_s, index_r \rangle$
$\Rightarrow Inv'$

PROOF:
LET: $p \in sSet$

$\langle 8\rangle 1$. $\wedge\ Inv$
$\wedge\ p[1] < index_r$
$\wedge\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\wedge$ UNCHANGED $\langle rSet, index_s, index_r \rangle$
$\Rightarrow Inv'$

PROOF:
$\langle 9\rangle 1$. $\wedge\ rSet \subseteq UpperSet(sSet, index_r)$
$\wedge\ p[1] < index_r$
$\wedge\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\wedge$ UNCHANGED $\langle rSet, index_r \rangle$
$\Rightarrow rSet' \subseteq UpperSet(sSet', index'_r)$

PROOF: From $p[1] < index_r$ follows
$UpperSet(sSet, index_r) \cap \{pair \in sSet : pair[1] \leq p[1]\} = \emptyset$. Now we obtain

$$\begin{aligned} rSet' &= rSet \\ &\subseteq UpperSet(sSet, index_r) \\ &= UpperSet(sSet, index_r) \setminus \{pair \in sSet : pair[1] \leq p[1]\} \\ &= UpperSet(sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}, index_r) \\ &= UpperSet(sSet', index'_r). \ \square \end{aligned}$$

$\langle 9\rangle 2$. $\wedge\ \vee\ sSet = \emptyset$
$\qquad \vee\ MaxIndex(sSet) < index_s$
$\wedge\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\wedge$ UNCHANGED $index_s$
$\Rightarrow \vee\ sSet' = \emptyset$
$\qquad \vee\ MaxIndex(sSet') < index'_s$

PROOF: From $sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$ follows $sSet' \subseteq sSet$.
From $sSet' \subseteq sSet$ follows $sSet' = \emptyset$ or
$MaxIndex(sSet') \leq MaxIndex(sSet) < index_s = index'_s$. $\square$

⟨9⟩3. $\land\ index_r \leq index_s$
$\land$ UNCHANGED $\langle index_s, index_r \rangle$
$\Rightarrow index'_r \leq index'_s$

PROOF: Trivial. □

⟨9⟩4. $\land\ \forall\, i \in Indices : |\{p \in sSet : p[1] = i\}| = 1$
$\land\ p[1] < index_r$
$\land\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\land$ UNCHANGED $\langle index_s, index_r \rangle$
$\Rightarrow \forall\, i \in Indices' : |\{p \in sSet' : p[1] = i\}| = 1$

PROOF:
LET: $SSubset = \{p \in sSet : p[1] \in Indices\}$

From $p[1] < index_r$ and the definition of *Indices* follows
$SSubset \cap \{pair \in sSet : pair[1] \leq p[1]\} = \emptyset$.
From this, the definition of *Indices*, $sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
and UNCHANGED $\langle index_s, index_r \rangle$ follows $SSubset' = SSubset$.
From UNCHANGED $\langle index_s, index_r \rangle$ follows $Indices' = Indices$.
Using this, $SSubset' = SSubset$ and the definition of *Indices*
we get the following equivalence:
$\forall\, i \in Indices : |\{p \in sSet : p[1] = i\}| = 1$
$\Leftrightarrow \forall\, i \in Indices : |\{p \in SSubset : p[1] = i\}| = 1$
$\Leftrightarrow \forall\, i \in Indices' : |\{p \in SSubset' : p[1] = i\}| = 1$
$\Leftrightarrow \forall\, i \in Indices' : |\{p \in sSet' : p[1] = i\}| = 1$. □

⟨9⟩5. Q.E.D.
PROOF: Propositional logic from ⟨9⟩1, ⟨9⟩2, ⟨9⟩3 and ⟨9⟩4. □

⟨8⟩2. Q.E.D.
PROOF: Predicate logic: forall-intro. □

⟨7⟩2. Q.E.D.
PROOF: Direct from the definition of *Purge*. □

⟨6⟩5. $\land\ Inv$
$\land$ UNCHANGED $Vars$
$\Rightarrow Inv'$

PROOF: Direct from the definitions of *Vars* and *Inv*. □

⟨6⟩6. Q.E.D.
PROOF: Propositional logic from ⟨6⟩1, ⟨6⟩2, ⟨6⟩3, ⟨6⟩4 and ⟨6⟩5. □

⟨5⟩2. Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. □

⟨4⟩2. Q.E.D.
PROOF: Direct from ⟨4⟩1 using rule INV1. □

⟨3⟩3. Q.E.D.

PROOF: Propositional logic from $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$. $\square$

$\langle 2 \rangle 2.$ $\land \Box Inv$
$\land \Box[SomeAction]_{Vars}$
$\Rightarrow \Box[HLS.SomeAction]_{HLS.Vars}$

PROOF:
$\langle 3 \rangle 1.$ $\land Inv$
$\land [SomeAction]_{Vars}$
$\Rightarrow [HLS.SomeAction]_{HLS.Vars}$

PROOF:
$\langle 4 \rangle 1.$ $\land Inv$
$\land \lor \exists a : Push(a)$
$\lor Pop$
$\lor Copy$
$\lor Purge$
$\lor$ UNCHANGED $Vars$
$\Rightarrow \lor \exists a : HLS.Push(a)$
$\lor HLS.Pop$
$\lor HLS.Transfer$
$\lor$ UNCHANGED $HLS.Vars$

PROOF:
$\langle 5 \rangle 1.$ $\land Inv$
$\land \exists a : Push(a)$
$\Rightarrow \exists a : HLS.Push(a)$

PROOF:
LET: a : CONSTANT

$\langle 6 \rangle 1.$ $\land Inv$
$\land Push(a)$
$\Rightarrow HLS.Push(a)$

PROOF:
$\langle 7 \rangle 1.$ $\land Inv$
$\land a \in Messages$
$\land \lor sSet = \emptyset$
$\lor index_s - MinIndex(sSet) < BufferLength$
$\land sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land index'_s = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$
$\Rightarrow \land a \in Messages$
$\land index_s - index_r < BufferLength$
$\land HLSsSet' = HLSsSet \cup \{\langle index_s, a \rangle\}$
$\land index'_s = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$

PROOF:
$\langle 8 \rangle 1.$ $a \in Messages \Rightarrow a \in Messages$

PROOF: Trivial. □

$\langle 8 \rangle 2.$ $\land$ Inv
$\land$ $\lor$ $sSet = \emptyset$
$\lor$ $index_s - MinIndex(sSet) < BufferLength$
$\Rightarrow index_s - index_r < BufferLength$

PROOF:

$\langle 9 \rangle 1.$ $\land$ Inv
$\land$ $sSet = \emptyset$
$\Rightarrow Indices = \emptyset$

PROOF: From $sSet = \emptyset$ follows $\forall\, i \,:\, |\{p \in sSet \,:\, p[1] = i\}| = 0$.
From this and $Inv.4$ follows $Indices = \emptyset$. □

$\langle 9 \rangle 2.$ $Indices = \emptyset \Rightarrow index_s - index_r < BufferLength$

PROOF: From $Indices = \emptyset$ follows $index_s \leq index_r$.
Using this and the refinement mapping we obtain
$index_s - index_r \leq 0 < BufferLength$. □

$\langle 9 \rangle 3.$ $\land$ $Indices \neq \emptyset$
$\land$ Inv
$\land$ $index_s - MinIndex(sSet) < BufferLength$
$\Rightarrow index_s - index_r < BufferLength$

PROOF: From $Indices \neq \emptyset$ and $Inv.4$ follows $|\{p \in sSet \,:\, p[1] = index_r\}| = 1$.
Then $MinIndex(sSet) \leq index_r$, and we obtain
$index_s - index_r \leq index_s - MinIndex(sSet) < BufferLength$. □

$\langle 9 \rangle 4.$ Q.E.D.
PROOF: Propositional logic from $\langle 9 \rangle 1$, $\langle 9 \rangle 2$ and $\langle 9 \rangle 3$. □

$\langle 8 \rangle 3.$ $\land$ Inv
$\land$ $sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$
$\Rightarrow HLSsSet' = HLSsSet \cup \{\langle index_s, a \rangle\}$

PROOF: From $Inv.1$ and $Inv.2$ follows $index_s \notin rSet$.
Using this, $Inv.3$ and the refinement mapping we obtain

$$\begin{aligned} HLSsSet' &= UpperSet(sSet', index_r') \setminus rSet' \\ &= UpperSet(sSet \cup \{\langle index_s, a \rangle\}, index_r) \setminus rSet \\ &= (UpperSet(sSet, index_r) \setminus rSet) \cup \{\langle index_s, a \rangle\} \\ &= HLSsSet \cup \{\langle index_s, a \rangle\}. \end{aligned}$$ □

$\langle 8 \rangle 4.$ $index_s' = index_s + 1 \Rightarrow index_s' = index_s + 1$

PROOF: Trivial. □

$\langle 8 \rangle 5.$ UNCHANGED $\langle rSet, index_r \rangle \Rightarrow$ UNCHANGED $\langle rSet, index_r \rangle$

PROOF: Trivial. □

$\langle 8 \rangle$6. Q.E.D.

PROOF: Propositional logic from $\langle 8 \rangle 1$, $\langle 8 \rangle 2$, $\langle 8 \rangle 3$, $\langle 8 \rangle 4$ and $\langle 8 \rangle 5$. $\square$

$\langle 7 \rangle$2. Q.E.D.

PROOF: Direct from the definitions of $Push(a)$ and $HLS.Push(a)$. $\square$

$\langle 6 \rangle$2. Q.E.D.

PROOF: Predicate logic: exist-intro and forall-intro. $\square$

$\langle 5 \rangle$2. $\land\ Inv$
$\land\ Pop$
$\Rightarrow\ HLS.Pop$

PROOF:

$\langle 6 \rangle$1. $\land\ Inv$
$\land\ \exists\, a\ :\ \land\ \langle index_r, a \rangle \in rSet$
$\qquad\qquad \land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\land\ index'_r = index_r + 1$
$\land\ \text{UNCHANGED}\ \langle sSet, index_s \rangle$
$\Rightarrow\ \land\ \exists\, a\ :\ \land\ \langle index_r, a \rangle \in rSet$
$\qquad\qquad \land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\quad \land\ index'_r = index_r + 1$
$\quad \land\ \text{UNCHANGED}\ \langle HLSsSet, index_s \rangle$

PROOF:

LET: a : CONSTANT

$\langle 7 \rangle$1. $\land\ Inv$
$\land\ \langle index_r, a \rangle \in rSet$
$\land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\land\ index'_r = index_r + 1$
$\land\ \text{UNCHANGED}\ \langle sSet, index_s \rangle$
$\Rightarrow\ \land\ \langle index_r, a \rangle \in rSet$
$\quad \land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\quad \land\ index'_r = index_r + 1$
$\quad \land\ \text{UNCHANGED}\ \langle HLSsSet, index_s \rangle$

PROOF:

$\langle 8 \rangle$1. $\langle index_r, a \rangle \in rSet \Rightarrow \langle index_r, a \rangle \in rSet$

PROOF: Trivial. $\square$

$\langle 8 \rangle$2. $rSet' = rSet \setminus \{\langle index_r, a \rangle\} \Rightarrow rSet' = rSet \setminus \{\langle index_r, a \rangle\}$

PROOF: Trivial. $\square$

$\langle 8 \rangle$3. $index'_r = index_r + 1 \Rightarrow index'_r = index_r + 1$

PROOF: Trivial. $\square$

$\langle 8 \rangle 4.$ $\wedge$ *Inv*
$\wedge$ $\langle index_r, a \rangle \in rSet$
$\wedge$ $rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\wedge$ $index'_r = index_r + 1$
$\wedge$ UNCHANGED $sSet$
$\Rightarrow$ UNCHANGED $HLSsSet$

PROOF: From $\langle index_r, a \rangle \in rSet$, $Inv.1$ and $Inv.4$ follows
$\{p \in sSet : p[1] = index_r\} = \{\langle index_r, a \rangle\}$.
Using this and the refinement mapping we obtain
$HLSsSet' = UpperSet(sSet', index'_r) \setminus rSet'$
$= UpperSet(sSet, index_r + 1) \setminus (rSet \setminus \{\langle index_r, a \rangle\})$
$= (UpperSet(sSet, index_r) \setminus \{p \in sSet : p[1] = index_r\}) \setminus (rSet \setminus \{\langle index_r, a \rangle\})$
$= (UpperSet(sSet, index_r) \setminus \{\langle index_r, a \rangle\}) \setminus (rSet \setminus \{\langle index_r, a \rangle\})$
$= UpperSet(sSet, index_r) \setminus rSet$
$= HLSsSet$. $\Box$

$\langle 8 \rangle 5.$ UNCHANGED $index_s \Rightarrow$ UNCHANGED $index_s$

PROOF: Trivial. $\Box$

$\langle 8 \rangle 6.$ Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$, $\langle 8 \rangle 2$, $\langle 8 \rangle 3$, $\langle 8 \rangle 4$ and $\langle 8 \rangle 5$. $\Box$

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. $\Box$

$\langle 6 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of Pop and $HLS.Pop$. $\Box$

$\langle 5 \rangle 3.$ *Copy*
$\Rightarrow$ $\vee$ $HLS.Transfer$
$\vee$ UNCHANGED $HLS.Vars$

PROOF:
$\langle 6 \rangle 1.$ $\wedge$ $\exists p :$ $\wedge$ $p \in sSet$
$\wedge$ $p[1] \geq index_r$
$\wedge$ $rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow$ $\vee$ $HLS.Transfer$
$\vee$ UNCHANGED $HLS.Vars$

PROOF:
LET: $p \in sSet$

$\langle 7 \rangle 1.$ $\wedge$ $p[1] \geq index_r$
$\wedge$ $rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow$ $\vee$ $HLS.Transfer$
$\vee$ UNCHANGED $HLS.Vars$

PROOF:

$\langle 8\rangle 1.$ $\wedge\ p[1] \geq index_r$
$\wedge\ p \notin rSet$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow HLS.Transfer$

PROOF:

$\langle 9\rangle 1.$ $\wedge\ p[1] \geq index_r$
$\wedge\ p \notin rSet$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_r \rangle$
$\Rightarrow \exists\, p\ :\ \wedge\ p \in HLSsSet$
$\wedge\ HLSsSet' = HLSsSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$

PROOF:

$\langle 10\rangle 1.$ $\wedge\ p[1] \geq index_r$
$\wedge\ p \notin rSet$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_r \rangle$
$\Rightarrow \wedge\ p \in HLSsSet$
$\wedge\ HLSsSet' = HLSsSet \setminus \{p\}$
$\wedge\ rSet' = rSet \cup \{p\}$

PROOF: $p \in HLSsSet$ follows from $p \in sSet$, $p[1] \geq index_r$, $p \notin rSet$ and the definition of $HLSsSet$.
Using the refinement mapping we obtain

$$\begin{aligned} HLSsSet' &= UpperSet(sSet', index'_r) \setminus rSet' \\ &= UpperSet(sSet, index_r) \setminus (rSet \cup \{p\}) \\ &= UpperSet(sSet, index_r) \setminus rSet \setminus \{p\} \\ &= HLSsSet \setminus \{p\}. \end{aligned}$$

$rSet' = rSet \cup \{p\}$ follows trivially from $rSet' = rSet \cup \{p\}$. □

$\langle 10\rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro. □

$\langle 9\rangle 2.$ UNCHANGED $\langle index_s, index_r \rangle \Rightarrow$ UNCHANGED $\langle index_s, index_r \rangle$

PROOF: Trivial. □

$\langle 9\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$ and $\langle 9\rangle 2$. □

$\langle 8\rangle 2.$ $\wedge\ p \in rSet$
$\wedge\ rSet' = rSet \cup \{p\}$
$\wedge$ UNCHANGED $\langle sSet, index_s, index_r \rangle$
$\Rightarrow$ UNCHANGED $HLS.Vars$

PROOF: From $p \in rSet$ and $rSet' = rSet \cup \{p\}$ follows UNCHANGED $rSet$.
UNCHANGED $HLS.Vars$ follows directly from UNCHANGED $\langle sSet, rSet, index_s, index_r \rangle$

and the definition of $HLS.Vars$. $\square$

$\langle 8\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 8\rangle 1$ and $\langle 8\rangle 2$. $\square$

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 6\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of $Copy$. $\square$

$\langle 5\rangle 4.$ $Purge \Rightarrow \text{UNCHANGED } HLS.Vars$

PROOF:

$\langle 6\rangle 1.$
$$\begin{array}{l} \land\ \exists\, p : \land\ p \in sSet \\ \qquad\qquad \land\ p[1] < index_r \\ \qquad\qquad \land\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\} \\ \land\ \text{UNCHANGED } \langle rSet, index_s, index_r \rangle \\ \Rightarrow \text{UNCHANGED } \langle HLSsSet, rSet, index_s, index_r \rangle \end{array}$$

PROOF:

$\langle 7\rangle 1.$
$$\begin{array}{l} \land\ \exists\, p : \land\ p \in sSet \\ \qquad\qquad \land\ p[1] < index_r \\ \qquad\qquad \land\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\} \\ \land\ \text{UNCHANGED } \langle rSet, index_r \rangle \\ \Rightarrow \text{UNCHANGED } HLSsSet \end{array}$$

PROOF:
LET: $p \in sSet$

$\langle 8\rangle 1.$
$$\begin{array}{l} \land\ p[1] < index_r \\ \land\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\} \\ \land\ \text{UNCHANGED } \langle rSet, index_r \rangle \\ \Rightarrow \text{UNCHANGED } HLSsSet \end{array}$$

PROOF: From $p[1] < index_r$ follows
$UpperSet(sSet, index_r) \cap \{pair \in sSet : pair[1] \leq p[1]\} = \emptyset$.
Using this and the refinement mapping we obtain
$$\begin{aligned} HLSsSet' &= UpperSet(sSet', index'_r) \setminus rSet' \\ &= UpperSet(sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}, index_r) \setminus rSet \\ &= UpperSet(sSet, index_r) \setminus \{pair \in sSet : pair[1] \leq p[1]\} \setminus rSet \\ &= UpperSet(sSet, index_r) \setminus rSet \\ &= HLSsSet. \ \square \end{aligned}$$

$\langle 8\rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. $\square$

$\langle 7\rangle 2.$ $\text{UNCHANGED } \langle rSet, index_s, index_r \rangle \Rightarrow \text{UNCHANGED } \langle rSet, index_s, index_r \rangle$

PROOF: Trivial. $\square$

$\langle 7\rangle 3.$ Q.E.D.

PROOF: Propositional logic from $\langle 7\rangle 1$ and $\langle 7\rangle 2$. $\square$

$\langle 6\rangle 2$. Q.E.D.
PROOF: Direct from the definitions of *Purge* and *HLS.Vars*. $\square$

$\langle 5\rangle 5$. UNCHANGED *Vars* $\Rightarrow$ UNCHANGED *HLS.Vars*

PROOF: Direct from the definitions of *Vars* and *HLS.Vars*. $\square$

$\langle 5\rangle 6$. Q.E.D.
PROOF: Propositional logic from $\langle 5\rangle 1$, $\langle 5\rangle 2$, $\langle 5\rangle 3$, $\langle 5\rangle 4$ and $\langle 5\rangle 5$. $\square$

$\langle 4\rangle 2$. Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. $\square$

$\langle 3\rangle 2$. Q.E.D.
PROOF: Proof by STL5 and STL4. $\square$

$\langle 2\rangle 3$. Q.E.D.
PROOF: Direct from $\langle 2\rangle 1$ and $\langle 2\rangle 2$ using propositional logic. $\square$

$\langle 1\rangle 3$. Q.E.D.
PROOF: Direct from $\langle 1\rangle 1$ and $\langle 1\rangle 2$ using propositional logic. $\square$

6 Sliding Window with Linear Arithmetic

Module V specifies the sliding-window-style message retransmission mechanism with linear arithmetic. We introduce full decoupling between sender and receiver by modeling communications as two one-way communication channels: a forward channel *fBuffer* for the messages and a backward channel *bBuffer* for acknowledgements. These are the only common variables of the sender and the receiver sides.

The communication channels are handled as if they might lose data: sender and receiver can send messages rsp. acknowledgements repeatedly; the sender keeps the sent messages until they have been acknowledged, and the receiver acknowledges only what he has already *Pop*'d.

Rather than *Copy*ing a pair from one side to the other, the pair is put into *fBuffer* with *SendMsg*, and read from there by *ReadMsg*. *fBuffer* contains either one index-message pair or the zero element $\langle 0, \perp_{Messages} \rangle$ which indicates "no message". *SendMsg* copies an arbitrary message from *sSet* into *fBuffer*, no matter if this overwrites an unread earlier message. Reading a message deletes it from *fBuffer* and puts it into *rSet*.

SendAck and *ReadAck* replace *IV.Purge*: *SendAck* copies $index_r$ into *bBuffer*. Thus *bBuffer* contains the index of the oldest message, that might not yet be *Pop*'d. It can be understood as meaning "precisely those messages older than this index are guaranteed to be *Pop*'d". *ReadAck* removes all messages from *sSet* with an index older (i.e. smaller) than that in *bBuffer*. In contrast to *fBuffer*, *bBuffer* is never reset to 0.

module V

EXTENDS *CommonDefinitions*

VARIABLES $sSet, rSet, index_r, index_s, fBuffer, bBuffer$

$$Vars \triangleq \langle sSet, rSet, index_r, index_s, fBuffer, bBuffer \rangle$$

$$\begin{aligned} Init \triangleq\ & \land\ index_r = 0 \\ & \land\ index_s = 0 \\ & \land\ sSet = \emptyset \\ & \land\ rSet = \emptyset \\ & \land\ fBuffer = \langle 0, \perp_{Messages} \rangle \\ & \land\ bBuffer = 0 \end{aligned}$$

$$\begin{aligned} Push(a) \triangleq\ & \land\ a \in Messages \\ & \land\ (sSet = \emptyset) \lor (index_s - MinIndex(sSet) < BufferLength) \\ & \land\ sSet' = sSet \cup \{\langle index_s, a \rangle\} \\ & \land\ index_s' = index_s + 1 \\ & \land\ \text{UNCHANGED } \langle rSet, index_r, fBuffer, bBuffer \rangle \end{aligned}$$

$$\begin{aligned} Pop \triangleq\ & \land\ \exists\, a : \land\ \langle index_r, a \rangle \in rSet \\ & \qquad\quad\ \land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\} \\ & \land\ index_r' = index_r + 1 \\ & \land\ \text{UNCHANGED } \langle sSet, index_s, fBuffer, bBuffer \rangle \end{aligned}$$

$$\begin{aligned} SendMsg \triangleq\ & \land\ \exists\, p : \land\ p \in sSet \\ & \qquad\quad\ \land\ fBuffer' = p \\ & \land\ \text{UNCHANGED } \langle index_s, index_r, sSet, rSet, bBuffer \rangle \end{aligned}$$

$$
\begin{array}{lcl}
\mathit{ReadMsg} & \triangleq & \land\ \mathit{fBuffer} \neq \langle 0, \bot_{\mathit{Messages}} \rangle \\
& & \land\ \mathit{fBuffer}[1] < \mathit{index}_r + \mathit{BufferLength} \\
& & \land\ \mathit{fBuffer}[1] \geq \mathit{index}_r \\
& & \land\ \mathit{rSet}' = \mathit{rSet} \cup \{\mathit{fBuffer}\} \\
& & \land\ \mathit{fBuffer}' = \langle 0, \bot_{\mathit{Messages}} \rangle \\
& & \land\ \text{UNCHANGED } \langle \mathit{index}_s, \mathit{index}_r, \mathit{sSet}, \mathit{bBuffer} \rangle \\
\mathit{SendAck} & \triangleq & \land\ \mathit{bBuffer}' = \mathit{index}_r \\
& & \land\ \text{UNCHANGED } \langle \mathit{index}_s, \mathit{index}_r, \mathit{sSet}, \mathit{rSet}, \mathit{fBuffer} \rangle \\
\mathit{ReadAck} & \triangleq & \land\ \mathit{sSet}' = \mathit{sSet} \setminus \{\mathit{pair} \in \mathit{sSet} : \mathit{pair}[1] < \mathit{bBuffer}\} \\
& & \land\ \text{UNCHANGED } \langle \mathit{index}_s, \mathit{index}_r, \mathit{rSet}, \mathit{fBuffer}, \mathit{bBuffer} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{SomeAction} & \triangleq & \lor\ \exists\, a : \mathit{Push}(a) \\
& & \lor\ \mathit{Pop} \\
& & \lor\ \mathit{SendMsg} \\
& & \lor\ \mathit{ReadMsg} \\
& & \lor\ \mathit{SendAck} \\
& & \lor\ \mathit{ReadAck} \\
\mathit{Safety} & \triangleq & \land\ \mathit{Init} \\
& & \land\ \Box[\mathit{SomeAction}]_{\mathit{Vars}}
\end{array}
$$

V.Safety implies IV.Safety

module *V implies IV*

EXTENDS V

$$HLS \triangleq \text{INSTANCE } IV \text{ WITH } \begin{array}{l} sSet \leftarrow sSet \\ rSet \leftarrow rSet \\ index_s \leftarrow index_s \\ index_r \leftarrow index_r \end{array}$$

$$PurgeSet \triangleq \{pair \in sSet : pair[1] < bBuffer\}$$

THEOREM $Safety \Rightarrow HLS.Safety$

$\langle 0 \rangle 4.$ $Safety \Rightarrow HLS.Safety$

PROOF:
$\langle 1 \rangle 1.$ $Init \Rightarrow HLS.Init$

PROOF:
$\langle 2 \rangle 1.$

$$\begin{array}{l} \wedge\ sSet = \emptyset \\ \wedge\ rSet = \emptyset \\ \wedge\ index_s = 0 \\ \wedge\ index_r = 0 \\ \wedge\ fBuffer = \langle 0, \perp_{Messages} \rangle \\ \wedge\ bBuffer = 0 \\ \Rightarrow \wedge\ sSet = \emptyset \\ \quad\ \wedge\ rSet = \emptyset \\ \quad\ \wedge\ index_s = 0 \\ \quad\ \wedge\ index_r = 0 \end{array}$$

PROOF: Immediate. □

$\langle 2 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of $Init$ and $HLS.Init$. □

$\langle 1 \rangle 2.$

$$\begin{array}{l} \wedge\ Init \\ \wedge\ \Box[SomeAction]_{Vars} \\ \Rightarrow \Box[HLS.SomeAction]_{HLS.Vars} \end{array}$$

PROOF: We use the following invariant:
LET: $Inv \triangleq$

$$\begin{array}{l} 1.\wedge \vee\ fBuffer = \langle 0, \perp_{Messages} \rangle \\ \quad\ \ \vee\ fBuffer[1] < index_r \\ \quad\ \ \vee\ fBuffer \in sSet \\ 2.\wedge\ bBuffer \leq index_r \end{array}$$

We use $Inv.1$ to prove that $ReadMsg$ simulates $HLS.Copy$.
We use $Inv.2$ to prove that $ReadAck$ simulates $HLS.Purge$ or UNCHANGED $HLS.Vars$, and to prove that $Inv.1$ is invariant under $ReadAck$.

$\langle 2 \rangle 1.\ \wedge\ \mathit{Init}$
$\quad \wedge\ \Box[\mathit{SomeAction}]_{\mathit{Vars}}$
$\quad \Rightarrow\ \Box\mathit{Inv}$

PROOF:
$\langle 3 \rangle 1.\ \mathit{Init} \Rightarrow \mathit{Inv}$

PROOF: $\mathit{Inv}.1$ follows from $\mathit{fBuffer} = \langle 0, \bot_{\mathit{Messages}} \rangle$.
$\mathit{Inv}.2$ follows from $\mathit{bBuffer} = 0$ and $\mathit{index}_r = 0$. □

$\langle 3 \rangle 2.\ \wedge\ \mathit{Inv}$
$\quad \wedge\ \Box[\mathit{SomeAction}]_{\mathit{Vars}}$
$\quad \Rightarrow\ \Box\mathit{Inv}$

PROOF:
$\langle 4 \rangle 1.\ \wedge\ \mathit{Inv}$
$\quad \wedge\ [\mathit{SomeAction}]_{\mathit{Vars}}$
$\quad \Rightarrow\ \mathit{Inv}'$

PROOF:
$\langle 5 \rangle 1.\ \wedge\ \mathit{Inv}$
$\quad \wedge\ \vee\ \exists\, a\ :\ \mathit{Push}(a)$
$\quad\quad \vee\ \mathit{Pop}$
$\quad\quad \vee\ \mathit{SendMsg}$
$\quad\quad \vee\ \mathit{ReadMsg}$
$\quad\quad \vee\ \mathit{SendAck}$
$\quad\quad \vee\ \mathit{ReadAck}$
$\quad\quad \vee\ \text{UNCHANGED}\ \mathit{Vars}$
$\quad \Rightarrow\ \mathit{Inv}'$

PROOF:
$\langle 6 \rangle 1.\ \wedge\ \mathit{Inv}$
$\quad \wedge\ \exists\, a\ :\ \mathit{Push}(a)$
$\quad \Rightarrow\ \mathit{Inv}'$

PROOF:
LET: $a\ :\ \text{CONSTANT}$

$\langle 7 \rangle 1.\ \wedge\ \mathit{Inv}$
$\quad \wedge\ \mathit{Push}(a)$
$\quad \Rightarrow\ \mathit{Inv}'$

PROOF:

$\langle 8\rangle 1.$ $\wedge\ Inv$
$\wedge\ a \in Messages$
$\wedge \vee\ sSet = \emptyset$
$\quad \vee\ index_s - MinIndex(sSet) < BufferLength$
$\wedge\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\wedge\ index'_s = index_s + 1$
$\wedge$ UNCHANGED $\langle rSet, index_r, fBuffer, bBuffer\rangle$
$\Rightarrow Inv'$

PROOF:

$\langle 9\rangle 1.$ $\wedge \vee\ fBuffer = \langle 0, \perp_{Messages}\rangle$
$\quad \vee\ fBuffer[1] < index_r$
$\quad \vee\ fBuffer \in sSet$
$\wedge\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\wedge$ UNCHANGED $\langle index_r, fBuffer\rangle$
$\Rightarrow \vee\ fBuffer' = \langle 0, \perp_{Messages}\rangle$
$\quad \vee\ fBuffer'[1] < index'_r$
$\quad \vee\ fBuffer' \in sSet'$

PROOF:

$\langle 10\rangle 1.$ $\wedge\ fBuffer = \langle 0, \perp_{Messages}\rangle$
$\wedge$ UNCHANGED $fBuffer$
$\Rightarrow fBuffer' = \langle 0, \perp_{Messages}\rangle$

PROOF: Trivial. □

$\langle 10\rangle 2.$ $\wedge\ fBuffer[1] < index_r$
$\wedge$ UNCHANGED $\langle index_r, fBuffer\rangle$
$\Rightarrow fBuffer'[1] < index'_r$

PROOF: Trivial. □

$\langle 10\rangle 3.$ $\wedge\ fBuffer \in sSet$
$\wedge\ sSet' = sSet \cup \{\langle index_s, a\rangle\}$
$\wedge$ UNCHANGED $fBuffer$
$\Rightarrow fBuffer' \in sSet'$

PROOF: From $sSet' = sSet \cup \{\langle index_s, a\rangle\}$ follows $sSet \subseteq sSet'$. Now $fBuffer' = fBuffer \in sSet \subseteq sSet'$. □

$\langle 10\rangle 4.$ Q.E.D.
PROOF: Propositional logic from $\langle 10\rangle 1$, $\langle 10\rangle 2$ and $\langle 10\rangle 3$. □

$\langle 9\rangle 2.$ $\wedge\ bBuffer \leq index_r$
$\wedge$ UNCHANGED $\langle index_r, bBuffer\rangle$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: Trivial. □

$\langle 9\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9\rangle 1$ and $\langle 9\rangle 2$. □

$\langle 8 \rangle 2.$ Q.E.D.

PROOF: Direct from the definition of $Push(a)$. □

$\langle 7 \rangle 2.$ Q.E.D.

PROOF: Predicate logic: forall-intro. □

$\langle 6 \rangle 2.$ $\land$ Inv
$\land$ Pop
$\Rightarrow$ Inv'

PROOF:

$\langle 7 \rangle 1.$ $\land$ Inv
$\land\ \exists\, a : \land\ \langle index_r, a \rangle \in rSet$
$\qquad\qquad \land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $\langle sSet, index_s, fBuffer, bBuffer \rangle$
$\Rightarrow$ Inv'

PROOF:

LET: a : CONSTANT

$\langle 8 \rangle 1.$ $\land$ Inv
$\land\ \langle index_r, a \rangle \in rSet$
$\land\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $\langle sSet, index_s, fBuffer, bBuffer \rangle$
$\Rightarrow$ Inv'

PROOF:

$\langle 9 \rangle 1.$ $\land\ \lor\ fBuffer = \langle 0, \perp_{Messages} \rangle$
$\qquad \lor\ fBuffer[1] < index_r$
$\qquad \lor\ fBuffer \in sSet$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $\langle sSet, fBuffer \rangle$
$\Rightarrow\ \lor\ fBuffer' = \langle 0, \perp_{Messages} \rangle$
$\qquad \lor\ fBuffer'[1] < index'_r$
$\qquad \lor\ fBuffer' \in sSet'$

PROOF:

$\langle 10 \rangle 1.$ $\land\ fBuffer = \langle 0, \perp_{Messages} \rangle$
$\land$ UNCHANGED $fBuffer$
$\Rightarrow\ fBuffer' = \langle 0, \perp_{Messages} \rangle$

PROOF: Trivial. □

$\langle 10 \rangle 2.$ $\land\ fBuffer[1] < index_r$
$\land\ index'_r = index_r + 1$
$\land$ UNCHANGED $fBuffer$
$\Rightarrow\ fBuffer'[1] < index'_r$

PROOF: From $index'_r = index_r + 1$ follows $index_r \leq index'_r$.

Now $fBuffer'[1] = fBuffer[1] < index_r \leq index'_r$. □

⟨10⟩3. $\wedge$ $fBuffer \in sSet$
$\wedge$ UNCHANGED $\langle sSet, fBuffer \rangle$
$\Rightarrow fBuffer' \in sSet'$

PROOF: Trivial. □

⟨10⟩4. Q.E.D.
PROOF: Propositional logic from ⟨10⟩1, ⟨10⟩2 and ⟨10⟩3. □

⟨9⟩2. $\wedge$ $bBuffer \leq index_r$
$\wedge$ $index'_r = index_r + 1$
$\wedge$ UNCHANGED $bBuffer$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: From $index'_r = index_r + 1$ follows $index_r \leq index'_r$.
Now $bBuffer' = bBuffer \leq index_r \leq index'_r$. □

⟨9⟩3. Q.E.D.
PROOF: Propositional logic from ⟨9⟩1 and ⟨9⟩2. □

⟨8⟩2. Q.E.D.
PROOF: Predicate logic: forall-intro. □

⟨7⟩2. Q.E.D.
PROOF: Direct from the definition of *Pop*. □

⟨6⟩3. $\wedge$ Inv
$\wedge$ $SendMsg$
$\Rightarrow Inv'$

PROOF:
⟨7⟩1. $\wedge$ Inv
$\wedge$ $\exists p : \wedge\ p \in sSet$
$\qquad\qquad \wedge\ fBuffer' = p$
$\wedge$ UNCHANGED $\langle sSet, rSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow Inv'$

PROOF:
LET: $p \in sSet$

⟨8⟩1. $\wedge$ Inv
$\wedge$ $fBuffer' = p$
$\wedge$ UNCHANGED $\langle sSet, rSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow Inv'$

PROOF:
⟨9⟩1. $\wedge$ $fBuffer' = p$
$\wedge$ UNCHANGED $sSet$
$\Rightarrow fBuffer' \in sSet'$

PROOF: From $fBuffer' = p$ and $p \in sSet$ follows $fBuffer' \in sSet$. From this and UNCHANGED $sSet$ follows $fBuffer' \in sSet'$. □

$\langle 9 \rangle 2.$ $\land\ bBuffer \leq index_r$
$\land$ UNCHANGED $\langle index_r, bBuffer \rangle$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: Trivial. □

$\langle 9 \rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 9 \rangle 1$ and $\langle 9 \rangle 2$. □

$\langle 8 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: forall-intro. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definition of *SendMsg*. □

$\langle 6 \rangle 4.$ $\land\ Inv$
$\land\ ReadMsg$
$\Rightarrow Inv'$

PROOF:
$\langle 7 \rangle 1.$ $\land\ Inv$
$\land\ fBuffer \neq \langle 0, \perp_{Messages} \rangle$
$\land\ fBuffer[1] < index_r + BufferLength$
$\land\ fBuffer[1] \geq index_r$
$\land\ rSet' = rSet \cup \{fBuffer\}$
$\land\ fBuffer' = \langle 0, \perp_{Messages} \rangle$
$\land$ UNCHANGED $\langle sSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow Inv'$

PROOF:
$\langle 8 \rangle 1.$ $fBuffer' = \langle 0, \perp_{Messages} \rangle \Rightarrow fBuffer' = \langle 0, \perp_{Messages} \rangle$

PROOF: Trivial. □

$\langle 8 \rangle 2.$ $\land\ bBuffer \leq index_r$
$\land$ UNCHANGED $\langle index_r, bBuffer \rangle$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: Trivial. □

$\langle 8 \rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definition of *ReadMsg*. □

$\langle 6 \rangle 5.$ $\land\ Inv$
$\land\ SendAck$
$\Rightarrow Inv'$

PROOF:

$\langle 7\rangle 1.$ $\land$ Inv
$\land$ $bBuffer' = index_r$
$\land$ UNCHANGED $\langle sSet, rSet, index_s, index_r, fBuffer\rangle$
$\Rightarrow Inv'$

PROOF:

$\langle 8\rangle 1.$ $\land \lor fBuffer = \langle 0, \bot_{Messages}\rangle$
$\lor fBuffer[1] < index_r$
$\lor fBuffer \in sSet$
$\land$ UNCHANGED $\langle sSet, index_r, fBuffer\rangle$
$\Rightarrow \lor fBuffer' = \langle 0, \bot_{Messages}\rangle$
$\lor fBuffer'[1] < index'_r$
$\lor fBuffer' \in sSet'$

PROOF: Trivial. □

$\langle 8\rangle 2.$ $\land$ $bBuffer' = index_r$
$\land$ UNCHANGED $index_r$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: Trivial. □

$\langle 8\rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 8\rangle 1$ and $\langle 8\rangle 2$. □

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Direct from the definition of $SendAck$. □

$\langle 6\rangle 6.$ $\land$ Inv
$\land$ $ReadAck$
$\Rightarrow Inv'$

PROOF:

$\langle 7\rangle 1.$ $\land$ Inv
$\land$ $sSet' = sSet \setminus PurgeSet$
$\land$ UNCHANGED $\langle rSet, index_s, index_r, fBuffer, bBuffer\rangle$
$\Rightarrow Inv'$

PROOF:

$\langle 8\rangle 1.$ $\land \lor fBuffer = \langle 0, \bot_{Messages}\rangle$
$\lor fBuffer[1] < index_r$
$\lor fBuffer \in sSet$
$\land$ $bBuffer \leq index_r$
$\land$ $sSet' = sSet \setminus PurgeSet$
$\land$ UNCHANGED $\langle index_r, fBuffer\rangle$
$\Rightarrow \lor fBuffer' = \langle 0, \bot_{Messages}\rangle$
$\lor fBuffer'[1] < index'_r$
$\lor fBuffer' \in sSet'$

PROOF:

$\langle 9 \rangle 1.$ $\wedge$ *fBuffer* $= \langle 0, \perp_{Messages} \rangle$
$\wedge$ UNCHANGED *fBuffer*
$\Rightarrow$ *fBuffer*$' = \langle 0, \perp_{Messages} \rangle$

PROOF: Trivial. □

$\langle 9 \rangle 2.$ $\wedge$ $fBuffer[1] < index_r$
$\wedge$ UNCHANGED $\langle index_r, fBuffer \rangle$
$\Rightarrow fBuffer'[1] < index'_r$

PROOF: Trivial. □

$\langle 9 \rangle 3.$ $\wedge$ $fBuffer[1] \geq index_r$
$\wedge$ $fBuffer \in sSet$
$\wedge$ $bBuffer \leq index_r$
$\wedge$ $sSet' = sSet \setminus PurgeSet$
$\wedge$ UNCHANGED *fBuffer*
$\Rightarrow fBuffer' \in sSet'$

PROOF: From $fBuffer[1] \geq index_r$ and $fBuffer \in sSet$ follows
$fBuffer \in \{p \in sSet : p[1] \geq index_r\}$.
From $bBuffer \leq index_r$ follows
$\{p \in sSet : p[1] \geq index_r\} \subseteq \{p \in sSet : p[1] \geq bBuffer\}$.
From $sSet' = sSet \setminus PurgeSet$ follows
$\{p \in sSet : p[1] \geq bBuffer\} = \{p \in sSet' : p[1] \geq bBuffer\}$.
Using these conclusions and UNCHANGED *fBuffer* we obtain

$$\begin{aligned} fBuffer' &= fBuffer \\ &\in \{p \in sSet : p[1] \geq index_r\} \\ &\subseteq \{p \in sSet : p[1] \geq bBuffer\} \\ &= \{p \in sSet' : p[1] \geq bBuffer\} \\ &\subseteq sSet'. \end{aligned}$$
□

$\langle 9 \rangle 4.$ Q.E.D.
PROOF: Propositional logic from $\langle 9 \rangle 1$, $\langle 9 \rangle 2$ and $\langle 9 \rangle 3$. □

$\langle 8 \rangle 2.$ $\wedge$ $bBuffer \leq index_r$
$\wedge$ UNCHANGED $\langle index_r, bBuffer \rangle$
$\Rightarrow bBuffer' \leq index'_r$

PROOF: Trivial. □

$\langle 8 \rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definition of *ReadAck*. □

$\langle 6 \rangle 7.$ $\wedge$ *Inv*
$\wedge$ UNCHANGED *Vars*
$\Rightarrow$ *Inv*$'$

PROOF: Direct from the definitions of $Vars$ and Inv. □

⟨6⟩8. Q.E.D.
PROOF: Propositional logic from ⟨6⟩1, ⟨6⟩2, ⟨6⟩3, ⟨6⟩4, ⟨6⟩5, ⟨6⟩6 and ⟨6⟩7. □

⟨5⟩2. Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. □

⟨4⟩2. Q.E.D.
PROOF: Direct from ⟨4⟩1 using rule INV1. □

⟨3⟩3. Q.E.D.
PROOF: Propositional logic from ⟨3⟩1 and ⟨3⟩2. □

⟨2⟩2. $\land \Box Inv$
$\land \Box[SomeAction]_{Vars}$
$\Rightarrow \Box[HLS.SomeAction]_{HLS.Vars}$

PROOF:
⟨3⟩1. $\land\ Inv$
$\land\ [SomeAction]_{Vars}$
$\Rightarrow [HLS.SomeAction]_{HLS.Vars}$

PROOF:
⟨4⟩1. $\land\ Inv$
$\land \lor \exists\, a : Push(a)$
$\lor\ Pop$
$\lor\ SendMsg$
$\lor\ ReadMsg$
$\lor\ SendAck$
$\lor\ ReadAck$
$\lor$ UNCHANGED $Vars$
$\Rightarrow \lor \exists\, a : HLS.Push(a)$
$\lor\ HLS.Pop$
$\lor\ HLS.Copy$
$\lor\ HLS.Purge$
$\lor$ UNCHANGED $HLS.Vars$

PROOF:
⟨5⟩1. $\exists\, a : Push(a) \Rightarrow \exists\, a : HLS.Push(a)$

PROOF:
LET: a : CONSTANT

⟨6⟩1. $Push(a) \Rightarrow HLS.Push(a)$

PROOF:

⟨7⟩1. $\land\ a \in Messages$
$\land \lor\ sSet = \emptyset$
$\lor\ index_s - MinIndex(sSet) < BufferLength$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land\ index_s' = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r, fBuffer, bBuffer \rangle$
$\Rightarrow \land\ a \in Messages$
$\land \lor\ sSet = \emptyset$
$\lor\ index_s - MinIndex(sSet) < BufferLength$
$\land\ sSet' = sSet \cup \{\langle index_s, a \rangle\}$
$\land\ index_s' = index_s + 1$
$\land$ UNCHANGED $\langle rSet, index_r \rangle$

PROOF:
⟨8⟩1. $a \in Messages \Rightarrow a \in Messages$

PROOF: Trivial. □

⟨8⟩2. $\lor\ sSet = \emptyset$
$\lor\ index_s - MinIndex(sSet) < BufferLength$
$\Rightarrow \lor\ sSet = \emptyset$
$\lor\ index_s - MinIndex(sSet) < BufferLength$

PROOF: Trivial. □

⟨8⟩3. $sSet' = sSet \cup \{\langle index_s, a \rangle\} \Rightarrow sSet' = sSet \cup \{\langle index_s, a \rangle\}$

PROOF: Trivial. □

⟨8⟩4. $index_s' = index_s + 1 \Rightarrow index_s' = index_s + 1$

PROOF: Trivial. □

⟨8⟩5. UNCHANGED $\langle rSet, index_r \rangle \Rightarrow$ UNCHANGED $\langle rSet, index_r \rangle$

PROOF: Trivial. □

⟨8⟩6. Q.E.D.
PROOF: Propositional logic from ⟨8⟩1, ⟨8⟩2, ⟨8⟩3, ⟨8⟩4 and ⟨8⟩5. □

⟨7⟩2. Q.E.D.
PROOF: Direct from the definitions of $Push(a)$ and $HLS.Push(a)$. □

⟨6⟩2. Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

⟨5⟩2. $Pop \Rightarrow HLS.Pop$

PROOF:

$\langle 6 \rangle 1.$ $\wedge\ \exists\, a : \wedge\ \langle index_r, a \rangle \in rSet$
$\quad\quad \wedge\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index'_r = index_r + 1$
$\wedge$ UNCHANGED $\langle sSet, index_s, fBuffer, bBuffer \rangle$
$\Rightarrow \wedge\ \exists\, a : \wedge\ \langle index_r, a \rangle \in rSet$
$\quad\quad \wedge\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\quad \wedge\ index'_r = index_r + 1$
$\quad \wedge$ UNCHANGED $\langle sSet, index_s \rangle$

PROOF:
LET: a : CONSTANT

$\langle 7 \rangle 1.$ $\wedge\ \langle index_r, a \rangle \in rSet$
$\wedge\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\wedge\ index'_r = index_r + 1$
$\wedge$ UNCHANGED $\langle sSet, index_s, fBuffer, bBuffer \rangle$
$\Rightarrow \wedge\ \langle index_r, a \rangle \in rSet$
$\quad \wedge\ rSet' = rSet \setminus \{\langle index_r, a \rangle\}$
$\quad \wedge\ index'_r = index_r + 1$
$\quad \wedge$ UNCHANGED $\langle sSet, index_s \rangle$

PROOF:
$\langle 8 \rangle 1.$ $\langle index_r, a \rangle \in rSet \Rightarrow \langle index_r, a \rangle \in rSet$

PROOF: Trivial. □

$\langle 8 \rangle 2.$ $rSet' = rSet \setminus \{\langle index_r, a \rangle\} \Rightarrow rSet' = rSet \setminus \{\langle index_r, a \rangle\}$

PROOF: Trivial. □

$\langle 8 \rangle 3.$ $index'_r = index_r + 1 \Rightarrow index'_r = index_r + 1$

PROOF: Trivial. □

$\langle 8 \rangle 4.$ UNCHANGED $\langle sSet, index_s \rangle \Rightarrow$ UNCHANGED $\langle sSet, index_s \rangle$

PROOF: Trivial. □

$\langle 8 \rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 8 \rangle 1$, $\langle 8 \rangle 2$, $\langle 8 \rangle 3$ and $\langle 8 \rangle 4$. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro and forall-intro. □

$\langle 6 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of Pop and $HLS.Pop$. □

$\langle 5 \rangle 3.$ $SendMsg \Rightarrow$ UNCHANGED $HLS.Vars$

PROOF:

$\langle 6 \rangle 1.$ $\wedge\ \exists\, p\ :\ \wedge\ p \in sSet$
$\qquad\qquad \wedge\ fBuffer' = p$
$\wedge\ \text{UNCHANGED}\ \langle sSet, rSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow\ \text{UNCHANGED}\ \langle sSet, rSet, index_s, index_r \rangle$

PROOF: Immediate. □

$\langle 6 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of *SendMsg* and *HLS.Vars*. □

$\langle 5 \rangle 4.$ $\wedge\ Inv$
$\wedge\ ReadMsg$
$\Rightarrow\ HLS.Copy$

PROOF:
$\langle 6 \rangle 1.$ $\wedge\ Inv$
$\wedge\ fBuffer \neq \langle 0, \perp_{Messages} \rangle$
$\wedge\ fBuffer[1] < index_r + BufferLength$
$\wedge\ fBuffer[1] \geq index_r$
$\wedge\ rSet' = rSet \cup \{fBuffer\}$
$\wedge\ fBuffer' = \langle 0, \perp_{Messages} \rangle$
$\wedge\ \text{UNCHANGED}\ \langle sSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow\ \wedge\ \exists\, p\ :\ \wedge\ p \in sSet$
$\qquad\qquad \wedge\ p[1] \geq index_r$
$\qquad\qquad \wedge\ rSet' = rSet \cup \{p\}$
$\quad \wedge\ \text{UNCHANGED}\ \langle sSet, index_s, index_r \rangle$

PROOF:
$\langle 7 \rangle 1.$ $\wedge\ Inv$
$\wedge\ fBuffer \neq \langle 0, \perp_{Messages} \rangle$
$\wedge\ fBuffer[1] < index_r + BufferLength$
$\wedge\ fBuffer[1] \geq index_r$
$\wedge\ rSet' = rSet \cup \{fBuffer\}$
$\wedge\ fBuffer' = \langle 0, \perp_{Messages} \rangle$
$\wedge\ \text{UNCHANGED}\ \langle sSet, index_s, index_r, bBuffer \rangle$
$\Rightarrow\ \wedge\ fBuffer \in sSet$
$\quad \wedge\ fBuffer[1] \geq index_r$
$\quad \wedge\ rSet' = rSet \cup \{fBuffer\}$
$\quad \wedge\ \text{UNCHANGED}\ \langle sSet, index_s, index_r \rangle$

PROOF:
$\langle 8 \rangle 1.$ $\wedge\ \vee\ fBuffer = \langle 0, \perp_{Messages} \rangle$
$\qquad \vee\ fBuffer[1] < index_r$
$\qquad \vee\ fBuffer \in sSet$
$\wedge\ fBuffer \neq \langle 0, \perp_{Messages} \rangle$
$\wedge\ fBuffer[1] \geq index_r$
$\Rightarrow\ fBuffer \in sSet$

PROOF: Simple propositional logic. □

$\langle 8 \rangle 2.$ $fBuffer[1] \geq index_r \Rightarrow fBuffer[1] \geq index_r$

PROOF: Trivial. $\square$

$\langle 8\rangle 3.$ $rSet' = rSet \cup \{fBuffer\} \Rightarrow rSet' = rSet \cup \{fBuffer\}$

PROOF: Trivial. $\square$

$\langle 8\rangle 4.$ $\text{UNCHANGED } \langle sSet, index_s, index_r \rangle \Rightarrow \text{UNCHANGED } \langle sSet, index_s, index_r \rangle$

PROOF: Trivial. $\square$

$\langle 8\rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 8\rangle 1$, $\langle 8\rangle 2$, $\langle 8\rangle 3$ and $\langle 8\rangle 4$. $\square$

$\langle 7\rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro. $\square$

$\langle 6\rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of *ReadMsg* and *HLS.Copy*. $\square$

$\langle 5\rangle 5.$ $SendAck \Rightarrow \text{UNCHANGED } HLS.Vars$

PROOF:
$\langle 6\rangle 1.$ $$\begin{array}{l} \wedge\ bBuffer' = index_r \\ \wedge\ \text{UNCHANGED } \langle sSet, rSet, index_s, index_r, fBuffer \rangle \\ \Rightarrow \text{UNCHANGED } \langle sSet, rSet, index_s, index_r \rangle \end{array}$$

PROOF: Immediate. $\square$

$\langle 6\rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of *SendAck* and *HLS.Vars*. $\square$

$\langle 5\rangle 6.$ $$\begin{array}{l} \wedge\ Inv \\ \wedge\ ReadAck \\ \Rightarrow \vee\ HLS.Purge \\ \quad\ \vee\ \text{UNCHANGED } HLS.Vars \end{array}$$

PROOF:
$\langle 6\rangle 1.$ $$\begin{array}{l} \wedge\ Inv \\ \wedge\ PurgeSet \neq \emptyset \\ \wedge\ ReadAck \\ \Rightarrow HLS.Transfer \end{array}$$

PROOF:
$\langle 7\rangle 1.$ $$\begin{array}{l} \wedge\ Inv \\ \wedge\ PurgeSet \neq \emptyset \\ \wedge\ sSet' = sSet \setminus PurgeSet \\ \wedge\ \text{UNCHANGED } \langle rSet, index_s, index_r, fBuffer, bBuffer \rangle \\ \Rightarrow \wedge\ \exists\, p : \wedge\ p \in sSet \\ \qquad\qquad\quad \wedge\ p[1] < index_r \\ \qquad\qquad\quad \wedge\ sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\} \\ \quad\ \wedge\ \text{UNCHANGED } \langle rSet, index_s, index_r \rangle \end{array}$$

PROOF:
LET: $p \triangleq MaxPair(PurgeSet)$

$\langle 8 \rangle 1.$ $\wedge$ Inv
$\wedge$ $PurgeSet \neq \emptyset$
$\wedge$ $sSet' = sSet \setminus PurgeSet$
$\wedge$ UNCHANGED $\langle rSet, index_s, index_r, fBuffer, bBuffer \rangle$
$\Rightarrow \wedge$ $p \in sSet$
$\wedge$ $p[1] < index_r$
$\wedge$ $sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$
$\wedge$ UNCHANGED $\langle rSet, index_s, index_r \rangle$

PROOF:
$\langle 9 \rangle 1.$ $PurgeSet \neq \emptyset \Rightarrow p \in sSet$

PROOF: From $PurgeSet \neq \emptyset$ follows that p is well defined.
From the definitions of p and $PurgeSet$ follows $p \in PurgeSet$ and $PurgeSet \subseteq sSet$, and thus $p \in sSet$. □

$\langle 9 \rangle 2.$ $\wedge$ $bBuffer \leq index_r$
$\wedge$ $PurgeSet \neq \emptyset$
$\Rightarrow p[1] < index_r$

PROOF: From $PurgeSet \neq \emptyset$ follows that p is well defined.
From the definitions of p and $PurgeSet$ follows $p[1] < bBuffer$.
From this and $bBuffer \leq index_r$ follows $p[1] < index_r$. □

$\langle 9 \rangle 3.$ $\wedge$ $PurgeSet \neq \emptyset$
$\wedge$ $sSet' = sSet \setminus PurgeSet$
$\Rightarrow sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$

PROOF: From $PurgeSet \neq \emptyset$ follows that p is well defined.
From the definitions of p and $PurgeSet$ follows
$\{pair \in sSet : pair[1] \leq p[1]\} = PurgeSet$.
From this and $sSet' = sSet \setminus PurgeSet$ follows
$sSet' = sSet \setminus \{pair \in sSet : pair[1] \leq p[1]\}$. □

$\langle 9 \rangle 4.$ UNCHANGED $\langle rSet, index_s, index_r \rangle \Rightarrow$ UNCHANGED $\langle rSet, index_s, index_r \rangle$

PROOF: Trivial. □

$\langle 9 \rangle 5.$ Q.E.D.
PROOF: Propositional logic from $\langle 9 \rangle 1$, $\langle 9 \rangle 2$, $\langle 9 \rangle 3$ and $\langle 9 \rangle 4$. □

$\langle 8 \rangle 2.$ Q.E.D.
PROOF: Predicate logic: exist-intro. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of $ReadAck$ and $HLS.Transfer$. □

$\langle 6 \rangle 2.$ $\wedge$ $PurgeSet = \emptyset$
$\wedge$ $ReadAck$

$\Rightarrow$ UNCHANGED $HLS.Vars$

PROOF:

$\langle 7 \rangle 1.$ $\land$ $PurgeSet = \emptyset$
$\land$ $sSet' = sSet \setminus PurgeSet$
$\land$ UNCHANGED $\langle rSet, index_s, index_r, fBuffer, bBuffer \rangle$
$\Rightarrow$ UNCHANGED $\langle sSet, rSet, index_s, index_r \rangle$

PROOF: UNCHANGED $\langle rSet, index_s, index_r \rangle$ follows immediately.
UNCHANGED $sSet$ follows from $PurgeSet = \emptyset$ and $sSet' = sSet \setminus PurgeSet$. □

$\langle 7 \rangle 2.$ Q.E.D.
PROOF: Direct from the definitions of $ReadAck$ and $HLS.Vars$. □

$\langle 6 \rangle 3.$ Q.E.D.
PROOF: Propositional logic from $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$. □

$\langle 5 \rangle 7.$ UNCHANGED $Vars \Rightarrow$ UNCHANGED $HLS.Vars$

PROOF: Direct from the definitions of $Vars$ and $HLS.Vars$. □

$\langle 5 \rangle 8.$ Q.E.D.
PROOF: Propositional logic from $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, $\langle 5 \rangle 3$, $\langle 5 \rangle 4$, $\langle 5 \rangle 5$, $\langle 5 \rangle 6$ and $\langle 5 \rangle 7$. □

$\langle 4 \rangle 2.$ Q.E.D.
PROOF: Propositional logic from TLA$^+$ definitions. □

$\langle 3 \rangle 2.$ Q.E.D.
PROOF: Proof by STL5 and STL4. □

$\langle 2 \rangle 3.$ Q.E.D.
PROOF: Direct from $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$ using propositional logic. □

$\langle 1 \rangle 3.$ Q.E.D.
PROOF: Direct from $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$ using propositional logic. □