A TLA Solution to the RPC-Memory Specification Problem —Memory Implementation Proof—

Martín Abadi, Leslie Lamport, and Stephan Merz

December 18, 1996

Abstract

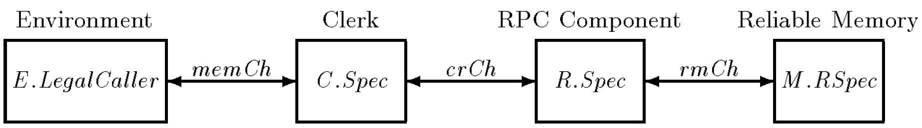
This document contains the complete proof of memory implementation whose lower levels were omitted in the printed version.

1 The Implementation Proof

We now formally assert that the composition of a memory clerk, an RPC component, and a reliable memory implements an unreliable memory; and we describe the proof of that assertion.

Since implementation is implication, the assertion that every behavior allowed by an implementation *Imp* satisfies a specification *Spec* is expressed by the formula $Imp \Rightarrow Spec$. However, we expect to prove the correctness of an implementation only under the assumption that the environment behaves correctly. If *Env* is the environment's specification, then we expect $Imp \Rightarrow Spec$ to be satisfied only by behaviors that satisfy *Env*. Thus, correctness of the implementation means that $Env \wedge Imp \Rightarrow Spec$ is valid. Composition is conjunction, so validity of this formula asserts that every behavior allowed by the composition of the environment and the implementation satisfies the specification.

The assertion that the composition of the clerk, RPC component, reliable memory, and environment specifications implies the unreliable memory's specification is theorem *Impl* of module *MemoryImplementation* in Figure 1. The specification of the unreliable memory's environment is formula *E.LegalCaller*, included from module *ProcedureInterface* by the imported *Memory* module. The composition is described schematically by the following picture.



module *MemoryImplementation*

import *MemoryParameters, Memory*

parameters

$crCh, rmCh$: VARIABLE

assumption

$FailureNotAValue \triangleq$ "RPCFailure" $\notin MemVals$

$Procs \triangleq \{$"Read", "Write"$\}$

$ArgNum \triangleq [i \in Procs \mapsto \textbf{case } (i = \text{"Read"}) \rightarrow 1,\ (i = \text{"Write"}) \rightarrow 2]$

include *RPC* **as** *R* **with** $sndCh \leftarrow crCh,\ rcvCh \leftarrow rmCh$

include *MemClerk* **as** *C* **with** $sndCh \leftarrow memCh,\ rcvCh \leftarrow crCh$

include *Memory* **as** *M* **with** $memCh \leftarrow rmCh$

theorem

$Impl \triangleq E.LegalCaller \wedge C.Spec \wedge R.Spec \wedge M.RSpec \Rightarrow USpec$

Figure 1: Module *MemoryImplementation*.

When composing two components by conjoining their specifications, the components are "connected" by instantiating their corresponding interface variable parameters with the same variable. The implementing module's specifications have been included with renaming; the specification *USpec* of the memory is imported from the *Memory* module.

The **theorem** statement asserts that the formula named *Impl* is a consequence of the assumptions *FailureNotAValue*[1] and *ParamAssump* (imported from module *MemoryParameters*), and the laws of TLA.

For convenience, we have gathered many of the definitions imported and included by module *MemoryImplementation* in Figure 2. In this figure and in our proof, we use the following naming conventions: (i) we eliminate the "*Inner.*" from symbol names—for example, writing $C.Retry(p)$ instead of $C.Inner.Retry(p)$, and (ii) if X is the name of a formula of the form $\forall p \in PrIds : Y$, then we let $X(p)$ denote the formula Y—as in $R.ISpec(p)$. The figure also defines the following additional symbols: pv, m, e, c, r, and $E.Next$.[2]

[1]We believe the theorem to be correct without assumption *FailureNotAValue*, but our proof uses the assumption.

[2]We define a number of operators with implicit parameters that are not parameters of module *MemoryImplementation*—for example, the parameters p and $result[p]$ that appear in the definition of

The Specification

Unreliable Memory Component (imported from *Memory*)

$$pv \triangleq pvars(p)$$

$$UNext(p) \triangleq Read(p) \lor (\exists\, l : Write(p, l)) \lor Return(p) \lor Fail(p)$$

$$\begin{aligned} UPSpec(p) \triangleq\ & \land\ PInit(p) \land \Box[UNext(p)]_{pv} \\ & \land\ \mathrm{WF}_{pv}(RNext(p)) \land \mathrm{WF}_{pv}(Return(p)) \end{aligned}$$

$$MSpec(l) \triangleq MInit(l) \land \Box[\exists\, p \in PrIds : Write(p, l)]_{mem[l]}$$

$$\begin{aligned} IUSpec \triangleq\ & \land\ \forall\, p \in PrIds : UPSpec(p) \\ & \land\ \forall\, l \in MemLocs : MSpec(l) \end{aligned}$$

$$USpec \triangleq \boldsymbol{\exists}\, mem, result : IUSpec$$

The Implementation

The Environment (included from *ProcedureInterface* via import of *Memory*)

$$e \triangleq E.caller(p)$$

$$E.Next(p) \triangleq \exists\, a \in LegalArgs : E.Call(p, a)$$

$$E.LegalCaller \triangleq \forall\, p \in PrIds : \neg E.Calling(p) \land \Box[E.Next(p)]_e$$

Clerk (included from *MemClerk*)

$$c \triangleq C.vars(p)$$

$$C.Next(p) \triangleq C.Forward(p) \lor C.Retry(p) \lor C.Reply(p)$$

$$\begin{aligned} C.ISpec(p) \triangleq\ & \land\ C.Init(p) \land \Box[C.Next(p)]_c \\ & \land\ \mathrm{WF}_c(C.Forward(p)) \land \mathrm{SF}_c(C.Reply(p)) \end{aligned}$$

$$C.Spec \triangleq \boldsymbol{\exists}\, cstate : \forall\, p \in PrIds : C.ISpec(p)$$

RPC Component (included from *RPC*)

$$r \triangleq R.vars(p)$$

$$R.Next(p) \triangleq R.Forward(p) \lor R.Reject(p) \lor R.Fail(p) \lor R.Reply(p)$$

$$R.ISpec(p) \triangleq R.Init(p) \land \Box[R.Next(p)]_r \land \mathrm{WF}_r(R.Next(p))$$

$$R.Spec \triangleq \boldsymbol{\exists}\, rstate : \forall\, p \in PrIds : R.ISpec(p)$$

Reliable Memory Component (included from *Memory*)

$$m \triangleq M.pvars(p)$$

$$M.RNext(p) \triangleq M.Read(p) \lor (\exists\, l : M.Write(p, l)) \lor M.Return(p)$$

$$\begin{aligned} M.RPSpec(p) \triangleq\ & \land\ M.PInit(p) \land \Box[M.RNext(p)]_m \\ & \land\ \mathrm{WF}_m(M.RNext(p)) \land \mathrm{WF}_m(M.Return(p)) \end{aligned}$$

$$M.MSpec(l) \triangleq M.MInit(l) \land \Box[\exists\, p \in PrIds : M.Write(p, l)]_{mem[l]}$$

$$\begin{aligned} M.IRSpec \triangleq\ & \land\ \forall\, p \in PrIds : M.RPSpec(p) \\ & \land\ \forall\, l \in MemLocs : M.MSpec(l) \end{aligned}$$

$$M.RSpec \triangleq \boldsymbol{\exists}\, mem, result : M.IRSpec$$

Figure 2: Formulas defined in module *MemoryImplementation*, plus a few extra definitions.

Theorem *Impl* has the form $F \Rightarrow \boldsymbol{\exists}\, mem, result : G$. In predicate logic, one proves a formula $P \Rightarrow \exists\, x : Q(x)$ by proving $P \Rightarrow Q(\overline{x})$ for a suitable instantiation $\overline{x}$ of x. In temporal logic, the instantiation is called a *refinement mapping* [1]. To prove *Impl*, we define a pair of state functions $\overline{mem}$ and $\overline{result}$ and prove $F \Rightarrow \overline{G}$, where $\overline{G}$ is the formula obtained by substituting $\overline{mem}$ and $\overline{result}$ for mem and $result$ in G.

For our proof, we let $\overline{mem}$ equal mem. To define $\overline{result}$, we must introduce a *history variable* [1]. Intuitively, a history variable a is one that is added to remember what happened in the past. Formally, proving $F \Rightarrow \overline{G}$ by "adding a history variable a" means choosing a variable a that does not appear in F and $\overline{G}$, finding a formula *Hist* of a particular form that guarantees that $\boldsymbol{\exists}\, a : Hist$ is valid, and proving $F \wedge Hist \Rightarrow \overline{G}$. Our history variable *rmhist* is defined so that, for each p in *PrIds*, the value of $rmhist[p]$ is initially equal to "A". It is set to "B" when process p of the reliable memory component returns to the RPC component or when process p of the RPC component issues a failure return to the clerk. It is reset to "A" when process p of the clerk returns to the caller. Formally, we define:

$$
\begin{array}{lcl}
h & \triangleq & rmhist[p] \\
HNext(p) & \triangleq & h' = \textbf{if } M.Return(p) \vee R.Fail(p) \\
 & & \qquad\quad \textbf{then } \text{"B"} \\
 & & \qquad\quad \textbf{else } \textbf{if } C.Reply(p) \textbf{ then } \text{"A"} \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \textbf{else } h \\
Hist & \triangleq & \forall\, p \in PrIds : (h = \text{"A"}) \wedge \Box[HNext(p)]_{\langle c,r,m,h\rangle}
\end{array}
$$

It should be intuitively obvious that, for every p in *PrIds*, formula *Hist* implies that the value of $rmhist[p]$ at any time is determined by the values of c, r, and m up to that time. A general theorem of TLA proves the validity of $\boldsymbol{\exists}\, rmhist : Hist$.

2 The High-Level Proof

We describe a structured proof of theorem *Impl*, in the style of [2]. We first present the high-level proof. It uses the state function $\overline{result}$, which we define later (the high-level proof is independent of its definition), and the temporal formula:

$$
\begin{array}{lcl}
IPImp(p) & \triangleq & E.LegalCaller(p) \wedge C.ISpec(p) \wedge R.ISpec(p) \\
 & & \quad \wedge\, M.RPSpec(p) \wedge (\forall\, l \in MemLocs : M.MSpec(l)) \wedge Hist(p)
\end{array}
$$

For any formula F, we let $\overline{F}$ be the formula obtained by substituting $\overline{result}$ for $result$ in F. Note that all formulas in the proof are interpreted in the context of the

m. If we were being truly formal, such definitions would occur in modules that made the parameters explicit, and these modules would then be included in the proof in contexts where the parameters are declared.

MemoryImplementation module. The variable declarations in the ASSUME (including the implicit declaration of p in the assumption $p \in PrIds$) are necessary, otherwise the formulas in the PROVE part would contain undeclared variables. The following high-level proof is a simple exercise in predicate-logic reasoning with the operators $\forall$ and $\exists$, since these operators (applied to temporal-logic formulas) obey the usual rules of first-order logic.

1. ASSUME: 1. $cstate, rstate, mem, result, rmhist$: VARIABLE
   2. $p \in PrIds$

   PROVE: $IPImp(p) \Rightarrow \overline{UPSpec(p)}$

   PROOF: Proved in section 3.1 below.
2. ASSUME: 1. $cstate, rstate, mem, result, rmhist$: VARIABLE
   2. $l \in MemLocs$

   PROVE: $(\forall\, q \in PrIds : IPImp(q)) \Rightarrow \overline{MSpec(l)}$

   PROOF: Proved in section 4 below.
3. ASSUME: $cstate, rstate, mem, result, rmhist$: VARIABLE

   PROVE: $E.LegalCaller \wedge C.ISpec \wedge R.ISpec \wedge M.IRSpec \wedge Hist \Rightarrow \overline{IUSpec}$

   PROOF: By steps 1 and 2, since $\forall$ distributes over $\wedge$, barring (which is just substitution) distributes over $\forall$ and $\wedge$, and we can deduce $(\forall\, u \in U : P(u)) \Rightarrow (\forall\, u \in U : Q(u))$ by proving $P(u) \Rightarrow Q(u)$ for any u in U.
4. ASSUME: $cstate, rstate, mem, result, rmhist$: VARIABLE

   PROVE: $E.LegalCaller \wedge C.ISpec \wedge R.ISpec \wedge M.IRSpec \wedge Hist \Rightarrow USpec$

   PROOF: By step 3, since we can deduce $F \Rightarrow \exists\, x : G(x)$ by proving $F \Rightarrow G(\overline{x})$, for some state function $\overline{x}$.
5. Q.E.D.

   PROOF: By step 4 and the validity of $\exists\, rmhist : Hist$, since we can deduce $(\exists\, x : F(x)) \Rightarrow G$ by proving $F(x) \Rightarrow G$, assuming x does not occur in G, and we can deduce the equivalence of $\exists\, x, y : F(x) \wedge G(y)$ and $(\exists\, x : F(x)) \wedge (\exists\, y : G(y))$, assuming x does not occur in $G(y)$ and y does not occur in $F(x)$.

3 The Lower-Level Proof

At the heart of our argument lie the proofs of steps 1 and 2. They are based on the predicate-action diagram of Figure 3. We introduce the abbreviations T and F for TRUE and FALSE, and UC for UNCHANGED. The operator S is defined to assert that

- For each of the three channels *memCh*, *crCh*, and *rmCh*, there is a call in progress on that channel iff the corresponding one of the first three arguments equals T.
- The values of $cstate[p]$, $rstate[p]$, and $rmhist[p]$ equal the last three arguments, where "AB" indicates a value of either "A" or "B".

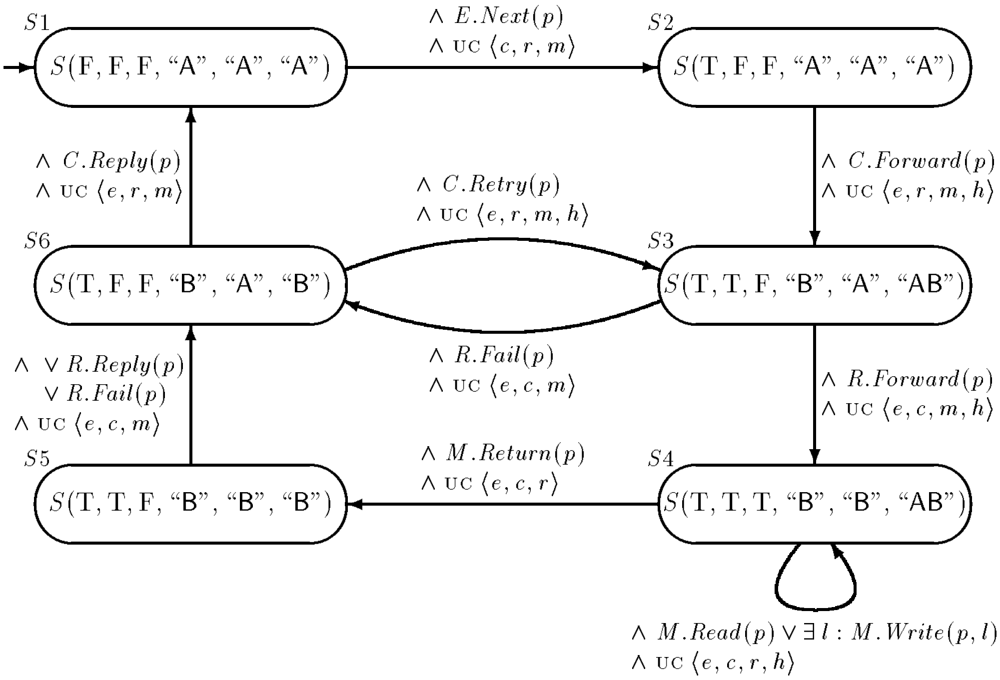


Figure 3: A predicate-action diagram of $\langle e, c, r, m, h \rangle$ for $IPImp(p)$, where p is an element of $PrIds$.

- Certain relations hold among the other variables—for example, if the first argument is T, then $memCh[p].arg$ is an element of $LegalArgs$.
- $mem[l]$ is an element of $MemVals$, for all l in $MemLocs$.

The formal definition of S appears in Figure 4. It may help in understanding this definition to observe that:

$$E.Calling(p) \equiv C.Snd.Calling(p)$$
$$C.Rcv.Calling(p) \equiv R.Snd.Calling(p)$$
$$R.Rcv.Calling(p) \equiv M.E.Calling(p)$$

We have labeled the state predicates in the predicate-action diagram $S1, \ldots, S6$. We define those labels to be synonymous with their respective predicates, so $S2$ equals $S(\mathrm{T}, \mathrm{F}, \mathrm{F}, \text{"A"}, \text{"A"}, \text{"A"})$. We define the state function $\overline{result}$ so that $\overline{result}[p]$ has the value given in Figure 4, for each p in $PrIds$.

3.1 The Proof of Step 1

Intuitively, the proof of step 1 is as follows.

$$
\begin{array}{l}
S(ECalling, CCalling, RCalling, cs, rs, rh) \;\triangleq \\
\quad \land \; \land \; ECalling \equiv E.Calling(p) \\
\quad \; \land \; ECalling \Rightarrow (memCh[p].arg \in LegalArgs) \\
\quad \land \; \land \; CCalling \equiv R.Snd.Calling(p) \\
\quad \; \land \; CCalling \Rightarrow (crCh[p].arg = C.RelayArg(p)) \\
\quad \; \land \; \neg CCalling \land (cstate[p] = \text{“B”}) \;\Rightarrow \\
\quad \qquad crCh[p].res \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\} \\
\quad \land \; \land \; RCalling \equiv M.E.Calling(p) \\
\quad \; \land \; RCalling \Rightarrow (rmCh[p].arg = R.RelayArg(p)) \\
\quad \; \land \; \neg RCalling \Rightarrow result[p] = NotAResult \\
\quad \; \land \; \neg RCalling \land (rstate[p] = \text{“B”}) \;\Rightarrow \\
\quad \qquad rmCh[p].res \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\} \\
\quad \land \; cs = cstate[p] \\
\quad \land \; rs = rstate[p] \\
\quad \land \; rmhist[p] \in \textbf{if } rh = \text{“AB”} \textbf{ then } \{\text{“A”}, \text{“B”}\} \\
\quad \phantom{\land \; rmhist[p] \in \textbf{if } rh = \text{“AB”}} \textbf{ else } \{rh\} \\
\quad \land \; result[p] \in MemVals \cup \{NotAResult, \text{“OK”}, \text{“BadArg”}\} \\
\quad \land \; \forall\, l \in MemLocs : mem[l] \in MemVals
\end{array}
$$

$$
\begin{array}{l}
\overline{result}[p] \; = \\
\quad \textbf{case } S1 \lor S2 \;\rightarrow\; result[p]\,, \\
\qquad\quad S3 \;\rightarrow\; \textbf{if } h = \text{“A”} \textbf{ then } result[p] \\
\qquad\qquad\qquad\qquad\qquad \textbf{else } \text{“MemFailure”}\,, \\
\qquad\quad S4 \;\rightarrow\; \textbf{if } (h = \text{“B”}) \land (result[p] = NotAResult) \\
\qquad\qquad\qquad \textbf{then } \text{“MemFailure”} \\
\qquad\qquad\qquad \textbf{else } result[p]\,, \\
\qquad\quad S5 \;\rightarrow\; rmCh[p].res\,, \\
\qquad\quad S6 \;\rightarrow\; \textbf{if } crCh[p].res = \text{“RPCFailure”} \textbf{ then } \text{“MemFailure”} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } crCh[p].res
\end{array}
$$

Figure 4: The formal definitions of S and $\overline{result}[p]$, for p in $PrIds$.

1.1. The implementation's initial condition implies the initial condition $S1$ of the predicate-action diagram.
1.2. The implementation's next-state action implies that the diagram describes all possible state transitions. There are six conditions, one for each state predicate in the diagram.
1.3. $\underline{\text{The initial}}$ $\underline{\text{condition}}$ $S1$ of the predicate-action diagram implies the initial condition $\overline{PInit(p)}$ of $\overline{UPSpec(p)}$.

1.4. Each of the actions allowed by the predicate-action diagram implements (implies) some disjunct of the next-state action $\overline{UNext(p)}$ of $\overline{UPSpec(p)}$, or else leaves $\overline{pv}$ unchanged.

1.5. All the temporal reasoning, including the proof of the fairness properties, is left for the final Q.E.D. step.

The formal proof makes use of several auxiliary lemmas, which are stated and proved in section 5. It is as follows:

1.1. $\neg E.Calling(p) \wedge C.Init(p) \wedge R.Init(p) \wedge M.PInit(p)$
$\quad \wedge (\forall\, l \in MemLocs : M.MInit(l)) \wedge (h = \text{“A”}) \Rightarrow S1$

PROOF: By propositional reasoning, it suffices to

ASSUME: 1. $\neg E.Calling(p)$
2. $C.Init(p)$
3. $R.Init(p)$
4. $M.PInit(p)$
5. $\forall\, l \in MemLocs : M.MInit(l)$
6. $h = \text{“A”}$

PROVE: $S(\mathrm{F}, \mathrm{F}, \mathrm{F}, \text{“A”}, \text{“A”}, \text{“A”})$

$\langle 3\rangle 1.$ $\mathrm{F} \equiv E.Calling(p)$
PROOF: Assumption 1·1.

$\langle 3\rangle 2.$ $\mathrm{F} \Rightarrow (memCh[p].arg \in LegalArgs)$
PROOF: Tautology.

$\langle 3\rangle 3.$ $\mathrm{F} \equiv R.Snd.Calling(p)$
PROOF: Assumption 1·2 implies $\neg R.Snd.Calling(p)$, since $C.Rcv.Calling(p) \equiv R.Snd.Calling(p)$.

$\langle 3\rangle 4.$ $\mathrm{F} \Rightarrow (crCh[p].arg = C.RelayArg(p))$
PROOF: Tautology.

$\langle 3\rangle 5.$ $\neg\mathrm{F} \wedge (cstate[p] = \text{“B”})$
$\quad \Rightarrow crCh[p].res \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$
PROOF: Assumption 1·2 implies $cstate[p] = \text{“A”}$.

$\langle 3\rangle 6.$ $\mathrm{F} \equiv M.E.Calling(p)$
PROOF: Assumption 1·3 and the definition of $R.Init(p)$ imply $\neg M.E.Calling(p)$, since $R.Rcv.Calling(p) \equiv M.E.Calling(p)$.

$\langle 3\rangle 7.$ $\mathrm{F} \Rightarrow (rmCh[p].arg = R.RelayArg(p))$
PROOF: Tautology.

$\langle 3\rangle 8.$ $\neg\mathrm{F} \Rightarrow (result[p] = NotAResult)$
PROOF: Assumption 1·4 implies $result[p] = NotAResult$.

$\langle 3\rangle 9.$ $\neg\mathrm{F} \wedge (rstate[p] = \text{“B”}) \Rightarrow rmCh[p].res \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$
PROOF: Assumption 1·3 implies $rstate[p] = \text{“A”}$.

$\langle 3\rangle 10.$ $\text{“A”} = cstate[p]$

PROOF: Assumption 1·2.

$\langle 3\rangle 11.$ "A" $= \mathit{rstate}[p]$

PROOF: Assumption 1·3.

$\langle 3\rangle 12.$ $\mathit{rmhist}[p] \in \textbf{if}$ "A" = "AB" $\textbf{then}$ {"A", "B"} $\textbf{else}$ {"A"}

PROOF: Assumption 1·6.

$\langle 3\rangle 13.$ $\mathit{result}[p] \in \mathit{MemVals} \cup \{\mathit{NotAResult}$, "OK", "BadArg"$\}$

PROOF: Assumption 1·4 implies $\mathit{result}[p] = \mathit{NotAResult}$.

$\langle 3\rangle 14.$ $\forall\, l \in \mathit{MemLocs} : \mathit{mem}[l] \in \mathit{MemVals}$

PROOF: Assumption 1·5 implies $\forall\, l \in \mathit{MemLocs} : \mathit{mem}[l] = \mathit{InitVal}$, from which the assertion follows by assumption *ParamAssump* of module *MemoryParameters*.

$\langle 3\rangle 15.$ Q.E.D.

PROOF: From steps $\langle 3\rangle 1$–$\langle 3\rangle 14$ by the definition of $S(\mathrm{F}, \mathrm{F}, \mathrm{F},$ "A", "A", "A"$)$.

1.2. ASSUME: 1. $\wedge\ [E.\mathit{Next}(p)]_e$
$\wedge\ [C.\mathit{Next}(p)]_c \wedge [R.\mathit{Next}(p)]_r \wedge [M.\mathit{RNext}(p)]_m$
$\wedge\ \forall\, l \in \mathit{MemLocs} : [\exists\, q \in \mathit{PrIds} : M.\mathit{Write}(q, l)]_{\mathit{mem}[l]}$
$\wedge\ [\mathit{HNext}(p)]_{\langle c, r, m, h\rangle}$
2. $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$

PROVE: 1. $S1 \Rightarrow S2' \wedge E.\mathit{Next}(p) \wedge \text{UC}\ \langle c, r, m\rangle$
2. $S2 \Rightarrow S3' \wedge C.\mathit{Forward}(p) \wedge \text{UC}\ \langle e, r, m, h\rangle$
3. $S3 \Rightarrow \vee\ S4' \wedge R.\mathit{Forward}(p) \wedge \text{UC}\ \langle e, c, m, h\rangle$
$\vee\ S6' \wedge R.\mathit{Fail}(p) \wedge \text{UC}\ \langle e, c, m\rangle$
4. $S4 \Rightarrow \vee\ S4' \wedge (M.\mathit{Read}(p) \vee \exists\, l : M.\mathit{Write}(p, l))$
$\wedge\ \text{UC}\ \langle e, c, r, h\rangle$
$\vee\ S5' \wedge M.\mathit{Return}(p) \wedge \text{UC}\ \langle e, c, r\rangle$
5. $S5 \Rightarrow S6' \wedge (R.\mathit{Reply}(p) \vee R.\mathit{Fail}(p)) \wedge \text{UC}\ \langle e, c, m\rangle$
6. $S6 \Rightarrow \vee\ S1' \wedge C.\mathit{Reply}(p) \wedge \text{UC}\ \langle e, r, m\rangle$
$\vee\ S3' \wedge C.\mathit{Retry}(p) \wedge \text{UC}\ \langle e, r, m, h\rangle$

$\langle 3\rangle 1.$ $M.\mathit{Return}(p) \Rightarrow (m' \neq m)$

PROOF: From the definition of $M.\mathit{Return}(p)$, which implies $\mathit{result}'[p] \neq \mathit{result}[p]$, and the definition of m by first-order logic.

$\langle 3\rangle 2.$ $R.\mathit{Fail}(p) \Rightarrow (r' \neq r)$

PROOF: It suffices to

ASSUME: $R.\mathit{Fail}(p)$

PROVE: $r' \neq r$

$\langle 4\rangle 1.$ $R.\mathit{Snd}.\mathit{Return}(p,$ "RPCFailure"$)$

PROOF: Assumption $\langle 3\rangle$

$\langle 4\rangle 2.$ $\mathit{crCh}[p].\mathit{cbit} \neq \mathit{crCh}[p].\mathit{rbit}$

PROOF: Step $\langle 4\rangle 1$ and the definitions of $R.\mathit{Snd}.\mathit{Return}(p, v)$ and $R.\mathit{Snd}.\mathit{Calling}(p)$.

$\langle 4\rangle 3.$ $\mathit{crCh}[p].\mathit{rbit}' = \mathit{crCh}[p].\mathit{cbit}$

PROOF: Step $\langle 4\rangle 1$ and the definition of $R.Snd.Return(p, v)$.

$\langle 4\rangle 4$. Q.E.D.

PROOF: Steps $\langle 4\rangle 2$ and $\langle 4\rangle 3$ imply

$crCh[p].rbit' \neq crCh[p].rbit$

from which the assertion follows by the definition of r.

$\langle 3\rangle 3$. $C.Reply(p) \Rightarrow (c' \neq c)$

PROOF: The proof is analogous to the proof of step $\langle 3\rangle 2$, since $C.Reply(p)$ implies $C.Snd.Return(p, ReplyVal(p))$.

$\langle 3\rangle 4$. ASSUME: $\forall\, l \in MemLocs : mem[l] \in MemVals$

PROVE: $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: By predicate logic, it suffices to

ASSUME: $l \in MemLocs$

PROVE: $mem'[l] \in MemVals$

$\langle 4\rangle 1$. CASE: $mem'[l] = mem[l]$

PROOF: Immediate by assumption $\langle 3\rangle$.

$\langle 4\rangle 2$. CASE: $(q \in PrIds) \wedge M.Write(q, l, v)$

$\langle 5\rangle 1$. CASE: $v \in MemVals$

PROOF: Assumptions $\langle 4\rangle$, $\langle 5\rangle$, and the definition of $M.Write(q, l, v)$ imply $mem'[l] = v$, and therefore $mem'[l] \in MemVals$.

$\langle 5\rangle 2$. CASE: $v \notin MemVals$

PROOF: Assumptions $\langle 4\rangle$ and $\langle 5\rangle$ and the definition of $M.Write(q, l, v)$ imply $mem'[l] = mem[l]$. The assertion follows by assumption $\langle 3\rangle$.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Steps $\langle 5\rangle 1$ and $\langle 5\rangle 2$

$\langle 4\rangle 3$. Q.E.D.

PROOF: Steps $\langle 4\rangle 1$ and $\langle 4\rangle 2$ and assumptions $\langle 3\rangle$ and 1·1, which implies $[\exists\, q \in PrIds : \exists\, v : M.Write(q, l, v)]_{mem[l]}$

$\langle 3\rangle 5$. ASSUME: $S1$

PROVE: 1. $S2'$

2. $E.Next(p)$

3. UNCHANGED $\langle c, r, m\rangle$

$\langle 4\rangle 1$. UNCHANGED c

$\langle 5\rangle 1$. $\neg E.Calling(p) \wedge (cstate[p] = \text{“A”})$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S1$.

$\langle 5\rangle 2$. $\neg C.Next(p)$

PROOF: From steps $\langle 5\rangle 1$ and Lemma 4 by propositional logic.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $C.Next(p) \vee$ UNCHANGED c.

$\langle 4\rangle 2$. UNCHANGED r

$\langle 5\rangle 1$. $\neg R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S1$.

$\langle 5 \rangle 2$. $\neg R.Next(p)$

PROOF: From steps $\langle 5 \rangle 1$ and Lemma 5 by propositional logic.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $R.Next(p) \vee \textsc{unchanged}\ r$.

$\langle 4 \rangle 3$. UNCHANGED m

$\langle 5 \rangle 1$. $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S1$.

$\langle 5 \rangle 2$. $\neg M.RNext(p)$

PROOF: From steps $\langle 5 \rangle 1$ and Lemma 6 by propositional logic.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $M.RNext(p) \vee \textsc{uc}\ m$.

$\langle 4 \rangle 4$. UNCHANGED h

PROOF: Steps $\langle 4 \rangle 1$–$\langle 4 \rangle 3$ and $\langle 3 \rangle 1$–$\langle 3 \rangle 3$ imply

$\neg M.Return(p) \wedge \neg R.Fail(p) \wedge \neg C.Reply(p)$

By the definition of $HNext(p)$ we conclude $HNext(p) \Rightarrow (h' = h)$, from which the assertion follows by assumption 1·1 and propositional reasoning.

$\langle 4 \rangle 5$. $E.Next(p)$

PROOF: Steps $\langle 4 \rangle 1$–$\langle 4 \rangle 4$ imply UNCHANGED $\langle c, r, m, h \rangle$, Using assumption 1·2, we obtain $e' \neq e$, and therefore $E.Next(p)$ by assumption 1·1.

$\langle 4 \rangle 6$. $S2'$

$\langle 5 \rangle 1$. $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6 \rangle 1$. $memCh[p].cbit' \neq memCh[p].rbit$

PROOF: Step $\langle 4 \rangle 5$ and the definitions of $E.Next(p)$ and $E.Call(p, v)$.

$\langle 6 \rangle 2$. $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4 \rangle 1$ and the definition of $C.vars(p)$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$ imply

$memCh[p].cbit' \neq memCh[p].rbit$

which by definition means $E.Calling(p)'$.

$\langle 5 \rangle 2$. $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Step $\langle 4 \rangle 5$ and the definition of $E.Next(p)$ imply

$\exists\, a \in LegalArgs : E.Call(p, a)$

The definition of $E.Call(p, a)$ and predicate logic yield

$\exists\, a \in LegalArgs : memCh[p].arg' = a$

which implies the assertion.

$\langle 5 \rangle 3$. $\mathrm{F} \equiv R.Snd.Calling(p)'$

PROOF: Assumption $\langle 3 \rangle$ implies $\neg R.Snd.Calling(p)$, by the definition of $S1$. The assertion follows by steps $\langle 4 \rangle 1$ and $\langle 4 \rangle 2$ and the definition of $R.Snd.Calling(p)'$.

$\langle 5 \rangle 4$. $\mathrm{F} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

PROOF: Tautology

$\langle 5\rangle 5.$ $\neg \mathrm{F} \wedge (cstate'[p] = \text{“B”})$
$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Assumption $\langle 3\rangle$ implies $cstate[p] = \text{“A”}$, by the definition of $S1$. Using step $\langle 4\rangle 1$, we conclude $cstate'[p] = \text{“A”}$, from which the assertion follows by propositional logic.

$\langle 5\rangle 6.$ $\mathrm{F} \equiv M.E.Calling(p)'$

PROOF: Assumption $\langle 3\rangle$ implies $\neg M.E.Calling(p)$, by the definition of $S1$. The assertion follows by steps $\langle 4\rangle 2$ and $\langle 4\rangle 3$ and the definitions of r, m, and $M.E.Calling(p)'$.

$\langle 5\rangle 7.$ $\mathrm{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology

$\langle 5\rangle 8.$ $\neg \mathrm{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption $\langle 3\rangle$ implies $result[p] = NotAResult$, by the definition of $S1$. The assertion follows by step $\langle 4\rangle 3$.

$\langle 5\rangle 9.$ $\neg \mathrm{F} \wedge (rstate'[p] = \text{“B”}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: Assumption $\langle 3\rangle$ implies $rstate[p] = \text{“A”}$, by the definition of $S1$. The assertion follows by step $\langle 4\rangle 2$.

$\langle 5\rangle 10.$ $\text{“A”} = cstate'[p]$

PROOF: Assumption $\langle 3\rangle$, the definition of $S1$, which implies $\text{“A”} = cstate[p]$, and step $\langle 4\rangle 1$.

$\langle 5\rangle 11.$ $\text{“A”} = rstate'[p]$

PROOF: Assumption $\langle 3\rangle$, the definition of $S1$, which implies $\text{“A”} = rstate[p]$, and step $\langle 4\rangle 2$.

$\langle 5\rangle 12.$ $rmhist'[p] \in \textbf{if } \text{“A”} = \text{“AB”} \textbf{ then } \{\text{“A”}, \text{“B”}\} \textbf{ else } \{\text{“A”}\}$

PROOF: Assumption $\langle 3\rangle$, the definition of $S1$, which implies $rmhist[p] = \text{“A”}$, and step $\langle 4\rangle 4$.

$\langle 5\rangle 13.$ $result'[p] \in MemVals \cup \{NotAResult, \text{“OK”}, \text{“BadArg”}\}$

PROOF: Step $\langle 5\rangle 8$.

$\langle 5\rangle 14.$ $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: Assumption $\langle 3\rangle$, the definition of $S1$, and step $\langle 3\rangle 4$.

$\langle 5\rangle 15.$ Q.E.D.

PROOF: From steps $\langle 5\rangle 1$–$\langle 5\rangle 14$ by the definition of $S2'$.

$\langle 4\rangle 7.$ Q.E.D.

PROOF: Steps $\langle 4\rangle 1$–$\langle 4\rangle 6$

$\langle 3\rangle 6.$ ASSUME: $S2$

PROVE: 1. $S3'$
2. $C.Forward(p)$
3. UNCHANGED $\langle e, r, m, h\rangle$

$\langle 4\rangle 1$. UNCHANGED e

$\langle 5\rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S2$.

$\langle 5\rangle 2$. $\neg E.Next(p)$

PROOF: Steps $\langle 5\rangle 1$ and Lemma 3.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $E.Next(p) \lor$ UNCHANGED e.

$\langle 4\rangle 2$. UNCHANGED r

$\langle 5\rangle 1$. $\neg R.Snd.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S2$.

$\langle 5\rangle 2$. $\neg R.Next(p)$

PROOF: From steps $\langle 5\rangle 1$ and Lemma 5 by propositional logic.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $R.Next(p) \lor$ UNCHANGED r.

$\langle 4\rangle 3$. UNCHANGED m

$\langle 5\rangle 1$. $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S2$.

$\langle 5\rangle 2$. $\neg M.RNext(p)$

PROOF: From steps $\langle 5\rangle 1$ and Lemma 6 by propositional logic.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $M.RNext(p) \lor$ UC m.

$\langle 4\rangle 4$. UNCHANGED h

PROOF: Steps $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 3\rangle 1$, and $\langle 3\rangle 2$ imply

$\neg M.Return(p) \land \neg R.Fail(p)$

Moreover, assumption $\langle 3\rangle$ and the definition of $S2$ imply $h =$ "A". By the definition of $HNext(p)$ we conclude $HNext(p) \Rightarrow (h' =$ "A"$)$, from which the assertion follows by assumption 1·1, which implies $HNext(p) \lor (h' = h)$ and equational reasoning.

$\langle 4\rangle 5$. $C.Forward(p)$

$\langle 5\rangle 1$. $C.Next(p)$

PROOF: Steps $\langle 4\rangle 1$–$\langle 4\rangle 4$ imply UNCHANGED $\langle e, r, m, h\rangle$. Using assumption 1·2, we conclude $c' \neq c$ and therefore $C.Next(p)$ by assumption 1·1.

$\langle 5\rangle 2$. $\neg C.Retry(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S2$ imply $cstate[p] =$ "A", whereas $C.Retry(p)$ implies $cstate[p] =$ "B".

$\langle 5\rangle 3$. $\neg C.Reply(p)$

PROOF: By the same argument as in the proof of step $\langle 5\rangle 2$.

$\langle 5\rangle 4$. Q.E.D.

PROOF: Step $\langle 5\rangle 1$ and the definition of $C.Next(p)$ imply

$C.Forward(p) \lor C.Retry(p) \lor C.Reply(p)$

The assertion follows by steps $\langle 5\rangle 2$ and $\langle 5\rangle 3$ and propositional reasoning.

$\langle 4\rangle 6.$ $S3'$

$\langle 5\rangle 1.$ $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6\rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S2$.

$\langle 6\rangle 2.$ $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6\rangle 3.$ $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4\rangle 5$ and the definitions of $C.Forward(p)$ and $C.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 2$ by the definition of $E.Calling(p)$.

$\langle 5\rangle 2.$ $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S2$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by steps $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5\rangle 3.$ $\mathrm{T} \equiv R.Snd.Calling(p)'$

$\langle 6\rangle 1.$ $crCh[p].cbit' \neq crCh[p].rbit$

PROOF: From step $\langle 4\rangle 5$, which by the definition of $C.Forward(p)$ implies $C.Rcv.Call(p, C.RelayArg(p))$, and the definition of $C.Rcv.Call(p, v)$.

$\langle 6\rangle 2.$ $crCh[p].rbit' = crCh[p].rbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of r and $R.Snd.rtrner(p)$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$ imply $crCh[p].cbit' \neq crCh[p].rbit'$, which by definition equals $R.Snd.Calling(p)'$.

$\langle 5\rangle 4.$ $\mathrm{T} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

$\langle 6\rangle 1.$ $crCh[p].arg' = C.RelayArg(p)$

PROOF: From step $\langle 4\rangle 5$, which by the definition of $C.Forward(p)$ implies $C.Rcv.Call(p, C.RelayArg(p))$, and the definition of $C.Rcv.Call(p, v)$.

$\langle 6\rangle 2.$ $C.RelayArg(p)' = C.RelayArg(p)$

PROOF: From step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$, we conclude UC $memCh[p].arg$. The assertion follows by the definition of $C.RelayArg(p)$, since $C.sndCh = memCh$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$.

$\langle 5\rangle 5.$ $\neg\mathrm{T} \wedge (cstate'[p] = \text{“B”})$

$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Tautology

$\langle 5\rangle 6.$ $\mathrm{F} \equiv M.E.Calling(p)'$

PROOF: Assumption $\langle 3\rangle$ and the definitio of $S2$ imply $\neg M.E.Calling(p)$. The assertion follows by steps $\langle 4\rangle 2$ and $\langle 4\rangle 3$ and the definitions of r, m, and $M.E.Calling(p)'$.

$\langle 5 \rangle 7.$ F $\Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology

$\langle 5 \rangle 8.$ $\neg$F $\Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S2$ imply $result[p] = NotAResult$. The assertion follows by step $\langle 4 \rangle 3$ and the definition of m.

$\langle 5 \rangle 9.$ $\neg$F $\wedge (rstate'[p] = \text{"B"}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}\}$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S2$ imply $rstate[p] = \text{"A"}$. The assertion follows by step $\langle 4 \rangle 2$, which implies UC $rstate[p]$.

$\langle 5 \rangle 10.$ $\text{"B"} = cstate'[p]$

PROOF: Step $\langle 4 \rangle 5$ and the definition of $C.Forward(p)$.

$\langle 5 \rangle 11.$ $\text{"A"} = rstate'[p]$

PROOF: Assumption $\langle 3 \rangle$, the definition of $S2$, and step $\langle 4 \rangle 2$.

$\langle 5 \rangle 12.$ $rmhist'[p] \in \textbf{if } \text{"AB"} = \text{"AB"} \textbf{ then } \{\text{"A"}, \text{"B"}\}$
$\qquad\qquad\qquad\qquad\qquad\qquad \textbf{else } \{\text{"AB"}\}$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S2$ imply $rmhist[p] = \text{"A"}$. The assertion follows by step $\langle 4 \rangle 4$.

$\langle 5 \rangle 13.$ $result'[p] \in MemVals \cup \{NotAResult, \text{"OK"}, \text{"BadArg"}\}$

PROOF: Step $\langle 5 \rangle 8$.

$\langle 5 \rangle 14.$ $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: Assumption $\langle 3 \rangle$, the definitions of $S2$ and S, and step $\langle 3 \rangle 4$.

$\langle 5 \rangle 15.$ Q.E.D.

PROOF: From steps $\langle 5 \rangle 1$–$\langle 5 \rangle 14$ by the definitions of $S3'$ and S.

$\langle 4 \rangle 7.$ Q.E.D.

PROOF: Steps $\langle 4 \rangle 1$–$\langle 4 \rangle 6$.

$\langle 3 \rangle 7.$ ASSUME: $S3$

PROVE: 1. $R.Forward(p) \vee R.Fail(p)$
2. $R.Forward(p) \Rightarrow S4' \wedge \text{UNCHANGED } \langle e, c, m, h \rangle$
3. $R.Fail(p) \Rightarrow S6' \wedge \text{UNCHANGED } \langle e, c, m \rangle$

$\langle 4 \rangle 1.$ UNCHANGED e

$\langle 5 \rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$.

$\langle 5 \rangle 2.$ $\neg E.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 3.

$\langle 5 \rangle 3.$ Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $E.Next(p) \vee \text{UNCHANGED } e$.

$\langle 4 \rangle 2.$ UNCHANGED c

$\langle 5 \rangle 1.$ $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$.

$\langle 5 \rangle 2.$ $\neg C.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 4.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $C.Next(p) \vee \text{UNCHANGED } c$.

$\langle 4\rangle 3$. UNCHANGED m

$\langle 5\rangle 1$. $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S3$.

$\langle 5\rangle 2$. $\neg M.RNext(p)$

PROOF: Steps $\langle 5\rangle 1$ and Lemma 6.

$\langle 5\rangle 3$. Q.E.D.

PROOF: Step $\langle 5\rangle 2$ and assumption 1·1, which implies $M.RNext(p) \vee \text{UC } m$.

$\langle 4\rangle 4$. $R.Forward(p) \vee R.Fail(p)$

$\langle 5\rangle 1$. $r' \neq r$

PROOF: By propositional logic, it suffices to

ASSUME: $r' = r$

PROVE: FALSE

$\langle 6\rangle 1$. $\neg M.Return(p)$

PROOF: Steps $\langle 4\rangle 3$ and $\langle 3\rangle 1$.

$\langle 6\rangle 2$. $\neg R.Fail(p)$

PROOF: Assumption $\langle 5\rangle$ and step $\langle 3\rangle 2$.

$\langle 6\rangle 3$. $\neg C.Reply(p)$

PROOF: Steps $\langle 4\rangle 2$ and $\langle 3\rangle 3$.

$\langle 6\rangle 4$. $h' = h$

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ and the definition of $HNext(p)$ imply $HNext(p) \Rightarrow h' = h$. The assertion follows by assumption 1·1, which implies $HNext(p) \vee \text{UNCHANGED } h$.

$\langle 6\rangle 5$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 6\rangle 4$, and assumption $\langle 5\rangle$, we obtain UNCHANGED $\langle e, c, r, m, h\rangle$ in contradiction to assumption 1·2.

$\langle 5\rangle 2$. $R.Next(p)$

PROOF: Step $\langle 5\rangle 1$ and assumption 1·1, which implies $R.Next(p) \vee (r' = r)r$.

$\langle 5\rangle 3$. $\neg R.Reject(p)$

$\langle 6\rangle 1$. $memCh[p].arg \in LegalArgs$

PROOF: Assumption $\langle 3\rangle$, the definition of $S3$, which implies $E.Calling(p)$, and the definition of S.

$\langle 6\rangle 2$. $R.RelayArg(p) = memCh[p].arg$

PROOF: From assumption $\langle 3\rangle$ and the definition of $S3$ by lemma 7.

$\langle 6\rangle 3$. Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$ imply $R.RelayArg(p) \in LegalArgs$, which by definition of $R.LegalRcvArgs$ means $R.RelayArg(p) \in R.LegalRcvArgs$ and implies $\neg R.Reject(p)$ by definition of $R.Reject(p)$.

$\langle 5\rangle 4$. $\neg R.Reply(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$ implies $rstate[p] = $ "A", contradicting $R.Reply(p)$, which implies $rstate[p] = $ "B".

$\langle 5 \rangle 5$. Q.E.D.

PROOF: Steps $\langle 5 \rangle 2$–$\langle 5 \rangle 4$ and the definition of $R.Next(p)$.

$\langle 4 \rangle 5$. ASSUME: $R.Forward(p)$
PROVE: $h' = h$

$\langle 5 \rangle 1$. $\neg M.Return(p)$

PROOF: Steps $\langle 4 \rangle 3$ and $\langle 3 \rangle 1$.

$\langle 5 \rangle 2$. $\neg R.Fail(p)$

PROOF: From assumption $\langle 4 \rangle$ and the definition of $R.Forward(p)$, we obtain UC $R.Snd.rtrner(p)$. On the other hand, the definitions of $R.Fail(p)$ and $R.Snd.Return(p,$ "RPCFailure"$)$ imply $crCh[p].rbit' \neq crCh[p].rbit$.

$\langle 5 \rangle 3$. $\neg C.Reply(p)$

PROOF: Steps $\langle 4 \rangle 2$ and $\langle 3 \rangle 3$.

$\langle 5 \rangle 4$. Q.E.D.

PROOF: Steps $\langle 5 \rangle 1$–$\langle 5 \rangle 3$ and the definition of $HNext$ imply $HNext(p) \Rightarrow h' = h$. The assertion follows by assumption 1·1, which implies $HNext(p) \vee (h' = h)$, and simple predicate logic.

$\langle 4 \rangle 6$. ASSUME: $R.Forward(p)$
PROVE: $S4'$

$\langle 5 \rangle 1$. $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6 \rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$.

$\langle 6 \rangle 2$. $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6 \rangle 3$. $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Snd.rtrner(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ by the definition of $E.Calling(p)$.

$\langle 5 \rangle 2$. $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3 \rangle$ and the definitions of $S3$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by steps $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5 \rangle 3$. $\mathrm{T} \equiv R.Snd.Calling(p)'$

$\langle 6 \rangle 1$. $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$.

$\langle 6 \rangle 2$. $crCh[p].cbit' = crCh[p].cbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6 \rangle 3$. $crCh[p].rbit' = crCh[p].rbit$

PROOF: From assumption $\langle 4 \rangle$, using the definitions of $R.Forward(p)$ and

$R.Snd.rtrner(p)$.

$\langle 6\rangle 4$. Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $R.Snd.Calling(p)$.

$\langle 5\rangle 4$. $\mathrm{T} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

$\langle 6\rangle 1$. $crCh[p].arg = C.RelayArg(p)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S3$ and S.

$\langle 6\rangle 2$. $crCh[p].arg' = crCh[p].arg$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6\rangle 3$. $C.RelayArg(p)' = C.RelayArg(p)$

PROOF: By step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$, we have UC $memCh[p].arg$. The assertion follows by the definition of $C.RelayArg(p)$, since $C.sndCh = memCh$.

$\langle 6\rangle 4$. Q.E.D.

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 3$.

$\langle 5\rangle 5$. $\neg\mathrm{T} \wedge (cstate'[p] = \text{“B”})$
$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Tautology

$\langle 5\rangle 6$. $\mathrm{T} \equiv M.E.Calling(p)'$

$\langle 6\rangle 1$. $rmCh[p].cbit' \neq rmCh[p].rbit$

PROOF: From assumption $\langle 4\rangle$ and the definitions of $R.Forward(p)$ and $R.Rcv.Call(p, v)$.

$\langle 6\rangle 2$. $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: Step $\langle 4\rangle 3$ and the definition of $M.E.rtrner(p)$.

$\langle 6\rangle 3$. Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$ imply $rmCh[p].cbit' \neq rmCh[p].rbit'$, which by definition equals $M.E.Calling(p)'$.

$\langle 5\rangle 7$. $\mathrm{T} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

$\langle 6\rangle 1$. $rmCh[p].arg' = R.RelayArg(p)$

PROOF: From assumption $\langle 4\rangle$ and the definitions of $R.Forward(p)$ and $R.Rcv.Call(p, R.RelayArg(p))$.

$\langle 6\rangle 2$. $R.RelayArg(p)' = R.RelayArg(p)$

PROOF: Step $\langle 4\rangle 2$ implies UC $R.sndCh[p]$, since $R.sndCh[p] = C.rcvCh[p] = crCh[p]$. The assertion follows by the definition of $R.RelayArg(p)$.

$\langle 6\rangle 3$. Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$.

$\langle 5\rangle 8$. $\neg\mathrm{T} \Rightarrow (result'[p] = NotAResult)$

PROOF: Tautology.

$\langle 5\rangle 9$. $\neg\mathrm{T} \wedge (rstate'[p] = \text{“B”}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: Tautology.

$\langle 5\rangle 10$. $\text{“B”} = cstate'[p]$

PROOF: From assumption $\langle 3 \rangle$ and the definition of $S3$, which implies $cstate[p] =$ "B" and step $\langle 4 \rangle 2$, which implies $cstate'[p] = cstate[p]$.

$\langle 5 \rangle 11$. "B" $= rstate'[p]$

PROOF: Assumption $\langle 4 \rangle$ and the definition of $R.Forward(p)$.

$\langle 5 \rangle 12$. $rmhist'[p] \in$ **if** "AB" = "AB" **then** {"A", "B"} **else** {"AB"}

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$, which implies $rmhist[p] \in$ {"A", "B"}, and step $\langle 4 \rangle 5$.

$\langle 5 \rangle 13$. $result'[p] \in MemVals \cup \{NotAResult,$ "OK", "BadArg"$\}$

PROOF: Assumption $\langle 3 \rangle$, the definitions of $S3$ and S, and step $\langle 4 \rangle 3$.

$\langle 5 \rangle 14$. $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: Assumption $\langle 3 \rangle$, the definitions of $S3$ and S, and step $\langle 3 \rangle 4$.

$\langle 5 \rangle 15$. Q.E.D.

PROOF: From steps $\langle 5 \rangle 1$–$\langle 5 \rangle 14$ by the definitions of $S4'$ and S'.

$\langle 4 \rangle 7$. ASSUME: $R.Fail(p)$
PROVE: $S6'$

$\langle 5 \rangle 1$. $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6 \rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S3$.

$\langle 6 \rangle 2$. $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6 \rangle 3$. $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Snd.rtrner(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ by the definition of $E.Calling(p)$.

$\langle 5 \rangle 2$. $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3 \rangle$ and the definitions of $S3$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by steps $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5 \rangle 3$. $\mathrm{F} \equiv R.Snd.Calling(p)'$

$\langle 6 \rangle 1$. $crCh[p].cbit' = crCh[p].cbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6 \rangle 2$. $crCh[p].rbit' = crCh[p].cbit$

PROOF: Assumption $\langle 4 \rangle$, the definitions of $R.Fail(p)$ and $R.Snd.Return(p, v)$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$ imply $crCh[p].rbit' = crCh[p].cbit'$, which by definition equals $\neg R.Snd.Calling(p)'$.

$\langle 5 \rangle 4$. $\mathrm{F} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

PROOF: Tautology

$\langle 5\rangle 5.$ $\neg \mathrm{F} \wedge (cstate'[p] = \text{“B”})$
$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: From assumption $\langle 4\rangle$ by expansion of the definitions of $R.Fail(p)$ and $R.Snd.Return(p, \text{“RPCFailure”})$, which imply $crCh[p].res' = \text{“RPCFailure”}$.

$\langle 5\rangle 6.$ $\mathrm{F} \equiv M.E.Calling(p)'$

$\langle 6\rangle 1.$ $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S3$.

$\langle 6\rangle 2.$ $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: Assumption $\langle 4\rangle$ and the definitions of $R.Fail(p)$ and $R.Rcv.caller(p)$.

$\langle 6\rangle 3.$ $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: Step $\langle 4\rangle 3$ and the definitions of m and $M.E.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $M.E.Calling(p)$.

$\langle 5\rangle 7.$ $\mathrm{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology.

$\langle 5\rangle 8.$ $\neg \mathrm{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S3$ and S imply $result[p] = NotAResult$. Step $\langle 4\rangle 3$ and the definition of m implies UC $result'[p]$.

$\langle 5\rangle 9.$ $\neg \mathrm{F} \wedge (rstate'[p] = \text{“B”}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: Assumption $\langle 4\rangle$ and the definition of $R.Fail(p)$ imply $rstate'[p] = \text{“A”}$.

$\langle 5\rangle 10.$ $\text{“B”} = cstate'[p]$

PROOF: From assumption $\langle 3\rangle$ and the definition of $S3$, which implies $cstate[p] = \text{“B”}$ and step $\langle 4\rangle 2$, which implies $cstate'[p] = cstate[p]$.

$\langle 5\rangle 11.$ $\text{“A”} = rstate'[p]$

PROOF: Assumption $\langle 4\rangle$ and the definition of $R.Fail(p)$.

$\langle 5\rangle 12.$ $rmhist'[p] \in \textbf{if } \text{“B”} = \text{“AB”} \textbf{ then } \{\text{“A”}, \text{“B”}\} \textbf{ else } \{\text{“B”}\}$

$\langle 6\rangle 1.$ $r' \neq r$

PROOF: Assumption $\langle 4\rangle$ and the definitions of $R.Fail(p)$, $R.Snd.Return(p, v)$, and $R.Snd.Calling(p)$ imply

$crCh[p].rbit' = crCh[p].cbit \neq crCh[p].rbit$

The assertion follows by definition of r, $R.vars(p)$, and $R.Snd.rtrner(p)$.

$\langle 6\rangle 2.$ $HNext(p)$

PROOF: Step $\langle 6\rangle 1$ and assumption 1·1 (which implies $HNext(p) \vee (r' = r)$)

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Step $\langle 6\rangle 2$, assumption $\langle 4\rangle$, and the definition of $HNext(p)$.

$\langle 5\rangle 13.$ $result'[p] \in MemVals \cup \{NotAResult, \text{“OK”}, \text{“BadArg”}\}$

PROOF: Step $\langle 5\rangle 8$.

$\langle 5\rangle 14.$ $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: From assumption $\langle 3\rangle$ and the definitions of $S3$ and S by step $\langle 3\rangle 4$.

$\langle 5 \rangle 15$. Q.E.D.

PROOF: From steps $\langle 5 \rangle 1$–$\langle 5 \rangle 14$ by the definitions of $S6'$ and S'.

$\langle 4 \rangle 8$. Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$–$\langle 4 \rangle 7$ by propositional logic.

$\langle 3 \rangle 8$. ASSUME: $S4$

PROVE: 1. $M.Read(p) \lor (\exists\, l : M.Write(p, l)) \lor M.Return(p)$

2. $M.Read(p) \Rightarrow S4' \land \text{UNCHANGED } \langle e, c, r, h \rangle$

3. $M.Write(p, l) \Rightarrow S4' \land \text{UNCHANGED } \langle e, c, r, h \rangle$

4. $M.Return(p) \Rightarrow S5' \land \text{UNCHANGED } \langle e, c, r \rangle$

$\langle 4 \rangle 1$. UNCHANGED e

$\langle 5 \rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S4$.

$\langle 5 \rangle 2$. $\neg E.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 3.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $E.Next(p) \lor \text{UNCHANGED } e$.

$\langle 4 \rangle 2$. UNCHANGED c

$\langle 5 \rangle 1$. $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S4$.

$\langle 5 \rangle 2$. $\neg C.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 4.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $C.Next \lor \text{UNCHANGED } c$.

$\langle 4 \rangle 3$. UNCHANGED r

$\langle 5 \rangle 1$. $M.E.Calling(p) \land (rstate[p] = \text{“B”})$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S4$.

$\langle 5 \rangle 2$. $\neg R.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 5.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $R.Next(p) \lor \text{UNCHANGED } r$.

$\langle 4 \rangle 4$. $\neg R.Fail(p)$

PROOF: Step $\langle 4 \rangle 3$ and step $\langle 3 \rangle 2$.

$\langle 4 \rangle 5$. $\neg C.Reply(p)$

PROOF: Steps $\langle 4 \rangle 2$ and $\langle 3 \rangle 3$.

$\langle 4 \rangle 6$. $M.RNext(p)$

$\langle 5 \rangle 1$. $m' \neq m$

PROOF: By propositional logic, it suffices to

ASSUME: $m' = m$

PROVE: FALSE

$\langle 6 \rangle 1$. $\neg M.Return(p)$

PROOF: Assumption $\langle 5\rangle$ and $\langle 3\rangle 1$.

$\langle 6\rangle 2.$ $h' = h$

PROOF: By steps $\langle 4\rangle 4$, $\langle 4\rangle 5$, and $\langle 6\rangle 1$, the definition of $HNext(p)$ implies $HNext(p) \Rightarrow h' = h$. The assertion follows by assumption 1·1, which implies $HNext(p) \vee (h' = h)$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: From steps $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 6\rangle 2$, and assumption $\langle 5\rangle$, we conclude UNCHANGED $\langle e, c, r, m, h\rangle$, in contradiction to assumption 1·2.

$\langle 5\rangle 2.$ Q.E.D.

PROOF: Step $\langle 5\rangle 1$ and Assumption 1·1, which implies $M.RNext(p) \vee (m' = m)$.

$\langle 4\rangle 7.$ ASSUME: $M.Read(p, l)$

PROVE: UNCHANGED h

$\langle 5\rangle 1.$ $\neg M.Return(p)$

PROOF: By definition, $M.Return(p)$ implies $M.E.Return(p, v)$. Using lemma 2, we conclude

$M.Return(p) \Rightarrow \neg$UNCHANGED $M.E.rtrner(p)$

The assertion follows by assumption $\langle 4\rangle$ and the definition of $M.Read(p, l)$, which implies UNCHANGED $M.E.rtrner(p)$.

$\langle 5\rangle 2.$ $HNext(p) \Rightarrow h' = h$

PROOF: From steps $\langle 4\rangle 4$, $\langle 4\rangle 5$, and $\langle 5\rangle 1$ by the the definition of $HNext(p)$.

$\langle 5\rangle 3.$ Q.E.D.

PROOF: From step $\langle 5\rangle 2$ and assumption 11, which implies $HNext(p) \vee h' = h$, by propositional logic.

$\langle 4\rangle 8.$ ASSUME: $M.Write(p, l, v)$

PROVE: UNCHANGED h

PROOF: Analogous to the proof of step $\langle 4\rangle 7$, since the definition of $M.Write(p, l, v)$ also implies UNCHANGED $M.E.rtrner(p)$.

$\langle 4\rangle 9.$ ASSUME: $M.Read(p, l)$

PROVE: $S4'$

$\langle 5\rangle 1.$ $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6\rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S4$.

$\langle 6\rangle 2.$ $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6\rangle 3.$ $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $E.Calling(p)$.

$\langle 5\rangle 2.$ $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S4$ and S imply $memCh[p].arg \in$

LegalArgs. The assertion follows by steps $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5 \rangle 3$. $\mathrm{T} \equiv R.Snd.Calling(p)'$

$\langle 6 \rangle 1$. $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S4$.

$\langle 6 \rangle 2$. $crCh[p].cbit' = crCh[p].cbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6 \rangle 3$. $crCh[p].rbit' = crCh[p].rbit$

PROOF: Step $\langle 4 \rangle 3$ and the definitions of r and $R.Snd.rtrner(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ by the definition of $R.Snd.Calling(p)$.

$\langle 5 \rangle 4$. $\mathrm{T} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

$\langle 6 \rangle 1$. $crCh[p].arg = C.RelayArg(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definitions of $S4$ and S.

$\langle 6 \rangle 2$. $crCh[p].arg' = crCh[p].arg$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6 \rangle 3$. $C.RelayArg(p)' = C.RelayArg(p)$

PROOF: Step $\langle 4 \rangle 1$ implies UC $memCh[p].arg$, by the definitions of e and $E.caller(p)$. The assertion follows by the definition of $C.RelayArg(p)$, since $C.sndCh = memCh$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$.

$\langle 5 \rangle 5$. $\neg \mathrm{T} \wedge (cstate'[p] = \text{“B”})$
$\quad \Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Tautology.

$\langle 5 \rangle 6$. $\mathrm{T} \equiv M.E.Calling(p)'$

$\langle 6 \rangle 1$. $M.E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S4$.

$\langle 6 \rangle 2$. $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: Step $\langle 4 \rangle 3$ and the definitions of r and $R.Snd.caller(p)$.

$\langle 6 \rangle 3$. $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: From assumption $\langle 4 \rangle$ and the definition of $M.Read(p)$, which implies UC $M.E.rtrner(p)$, by the definition of $M.E.rtrner(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ by the definition of $M.E.Calling(p)$.

$\langle 5 \rangle 7$. $\mathrm{T} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

$\langle 6 \rangle 1$. $rmCh[p].arg = R.RelayArg(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definitions of $S4$ and S.

$\langle 6 \rangle 2$. $rmCh[p].arg' = rmCh[p].arg$

PROOF: Step $\langle 4 \rangle 3$ and the definitions of r and $R.Rcv.caller(p)$.

$\langle 6\rangle 3.$ $R.RelayArg(p)' = R.RelayArg(p)$

PROOF: Step $\langle 4\rangle 2$ and the definition of c imply UC $crCh[p].arg$, from which the assertion follows by the definition of $R.RelayArg(p)$, since $R.sndCh = crCh$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 3$.

$\langle 5\rangle 8.$ $\neg \mathrm{T} \Rightarrow (result'[p] = NotAResult)$

PROOF: Tautology.

$\langle 5\rangle 9.$ $\neg \mathrm{T} \wedge (rstate'[p] = \text{"B"}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}\}$

PROOF: Tautology.

$\langle 5\rangle 10.$ $\text{"B"} = cstate'[p]$

PROOF: From assumption $\langle 3\rangle$ and the definition of $S4$, which implies $cstate[p] = \text{"B"}$, and step $\langle 4\rangle 2$, which implies $cstate'[p] = cstate[p]$.

$\langle 5\rangle 11.$ $\text{"B"} = rstate'[p]$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S4$ imply $rstate[p] = \text{"B"}$. The assertion follows by step $\langle 4\rangle 3$, which implies $rstate'[p] = rstate[p]$.

$\langle 5\rangle 12.$ $rmhist'[p] \in \textbf{if } \text{"AB"} = \text{"AB"} \textbf{ then } \{\text{"A"}, \text{"B"}\} \textbf{ else } \{\text{"AB"}\}$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S4$ imply $rmhist[p] \in \{\text{"A"}, \text{"B"}\}$. The assertion follows by step $\langle 4\rangle 7$ and assumption $\langle 4\rangle$.

$\langle 5\rangle 13.$ $result'[p] \in MemVals \cup \{NotAResult, \text{"OK"}, \text{"BadArg"}\}$

$\langle 6\rangle 1.$ CASE: $l \in MemLocs$

PROOF: Assumption $\langle 4\rangle$, the definition of $M.Read(p)$, and the case assumption imply $result'[p] = mem[l]$. The assertion follows by assumption $\langle 3\rangle$ and the definitions of $S4$ and S, which together with the case assumption imply $mem[l] \in MemVals$.

$\langle 6\rangle 2.$ CASE: $\neg(l \in MemLocs)$

PROOF: Assumption $\langle 4\rangle$, the definition of $M.Read(p)$, and the case assumption imply $result'[p] = \text{"BadArg"}$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$.

$\langle 5\rangle 14.$ $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: From assumption $\langle 3\rangle$ and the definitions of $S4$ and S by step $\langle 3\rangle 4$.

$\langle 5\rangle 15.$ Q.E.D.

PROOF: From steps $\langle 5\rangle 1$–$\langle 5\rangle 14$ by the definitions of $S4'$ and S'.

$\langle 4\rangle 10.$ ASSUME: $M.Write(p, l, v)$

PROVE: $S4'$

PROOF: Similar to the proof of step $\langle 4\rangle 9$, using the definition of $M.Write(p, l, v)$ instead of $M.Read(p, l)$.

$\langle 4\rangle 11.$ ASSUME: $M.Return(p)$

PROVE: $S5'$

$\langle 5\rangle 1.$ $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6\rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S4$.

$\langle 6\rangle 2.$ $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6\rangle 3.$ $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $E.Calling(p)$.

$\langle 5\rangle 2.$ $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S4$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by steps $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5\rangle 3.$ $\mathrm{T} \equiv R.Snd.Calling(p)'$

$\langle 6\rangle 1.$ $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S4$.

$\langle 6\rangle 2.$ $crCh[p].cbit' = crCh[p].cbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6\rangle 3.$ $crCh[p].rbit' = crCh[p].rbit$

PROOF: Step $\langle 4\rangle 3$ and the definitions of r and $R.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $R.Snd.Calling(p)$.

$\langle 5\rangle 4.$ $\mathrm{T} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

$\langle 6\rangle 1.$ $crCh[p].arg = C.RelayArg(p)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S4$ and S.

$\langle 6\rangle 2.$ $crCh[p].arg' = crCh[p].arg$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6\rangle 3.$ $C.RelayArg(p)' = C.RelayArg(p)$

PROOF: Step $\langle 4\rangle 1$ and the definition of e imply UC $memCh[p].arg$, from which the assertion follows by the definition of $C.RelayArg(p)$, since $C.sndCh = memCh$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 3$.

$\langle 5\rangle 5.$ $\neg\mathrm{T} \wedge (cstate'[p] = \text{“B”})$

$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Tautology.

$\langle 5\rangle 6.$ $\mathrm{F} \equiv M.E.Calling(p)'$

$\langle 6\rangle 1.$ $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: Step $\langle 4\rangle 3$ and the definitions of r and $R.Snd.caller(p)$.

$\langle 6\rangle 2.$ $rmCh[p].rbit' = rmCh[p].cbit$

PROOF: From assumption $\langle 4 \rangle$, using the definitions of $M.Return(p)$ and $M.E.Return(p, v)$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$ imply $rmCh[p].cbit' = rmCh[p].rbit'$, which by definition equals $\neg M.E.Calling(p)$.

$\langle 5 \rangle 7$. $\mathrm{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology.

$\langle 5 \rangle 8$. $\neg \mathrm{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption $\langle 4 \rangle$ and the definition of $M.Return(p)$.

$\langle 5 \rangle 9$. $\neg \mathrm{F} \land (rstate'[p] = \text{"B"}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}\}$

$\langle 6 \rangle 1$. $rmCh[p].res' = result[p]$

PROOF: From assumption $\langle 4 \rangle$, using the definitions of $M.Return(p)$ and $M.E.Return(p, result[p])$.

$\langle 6 \rangle 2$. $result[p] \in MemVals \cup \{NotAResult, \text{"OK"}, \text{"BadArg"}\}$

PROOF: Assumption $\langle 3 \rangle$ and the definitions of $S4$ and S.

$\langle 6 \rangle 3$. $result[p] \neq NotAResult$

PROOF: Assumption $\langle 4 \rangle$ and the definition of $M.E.Return(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ by simple predicate logic.

$\langle 5 \rangle 10$. $\text{"B"} = cstate'[p]$

PROOF: From assumption $\langle 3 \rangle$ and the definition of $S4$, which implies $cstate[p] = \text{"B"}$, and step $\langle 4 \rangle 2$, which implies $cstate'[p] = cstate[p]$.

$\langle 5 \rangle 11$. $\text{"B"} = rstate'[p]$

PROOF: Assumption $\langle 3 \rangle$and the definition of $S4$, which implies $rstate[p] = \text{"B"}$, and step $\langle 4 \rangle 3$, which implies $rstate'[p] = rstate[p]$.

$\langle 5 \rangle 12$. $rmhist'[p] \in \textbf{if } \text{"B"} = \text{"AB"} \textbf{ then } \{\text{"A"}, \text{"B"}\} \textbf{ else } \{\text{"B"}\}$

$\langle 6 \rangle 1$. $m' \neq m$

PROOF: Assumption $\langle 4 \rangle$ and the definition of $M.Return(p)$ imply $result'[p] \neq result[p]$. Therefore, the assertion follows from the definitions of m and $M.E.pvars(p)$.

$\langle 6 \rangle 2$. $HNext(p)$

PROOF: From step $\langle 6 \rangle 1$ and assumption 1·1, which implies $HNext(p) \lor (m' = m)$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Step $\langle 6 \rangle 2$, assumption $\langle 4 \rangle$, and the definition of $HNext(p)$ imply $rmhist'[p] = \text{"B"}$.

$\langle 5 \rangle 13$. $result'[p] \in MemVals \cup \{NotAResult, \text{"OK"}, \text{"BadArg"}\}$

PROOF: Step $\langle 5 \rangle 8$.

$\langle 5 \rangle 14$. $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: From assumption $\langle 3 \rangle$ and the definitions of $S4$ and S by step $\langle 3 \rangle 4$.

$\langle 5 \rangle 15$. Q.E.D.

PROOF: From steps $\langle 5 \rangle 1$–$\langle 5 \rangle 14$ by the definitions of $S5'$ and S'.

$\langle 4 \rangle 12$. Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$–$\langle 4 \rangle 3$ and $\langle 4 \rangle 6$–$\langle 4 \rangle 11$ by the definition of $M.RNext(p)$ and propositional logic.

$\langle 3 \rangle 9$. ASSUME: $S5$

PROVE: 1. $S6'$

2. $R.Reply(p) \lor R.Fail(p)$

3. UNCHANGED $\langle e, c, m \rangle$

$\langle 4 \rangle 1$. UNCHANGED e

$\langle 5 \rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S5$.

$\langle 5 \rangle 2$. $\neg E.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 3.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $E.Next(p) \lor$ UC e.

$\langle 4 \rangle 2$. UNCHANGED c

$\langle 5 \rangle 1$. $R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S5$.

$\langle 5 \rangle 2$. $\neg C.Next(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 4.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $C.Next(p) \lor$ UC c.

$\langle 4 \rangle 3$. UNCHANGED m

$\langle 5 \rangle 1$. $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S5$.

$\langle 5 \rangle 2$. $\neg M.RNext(p)$

PROOF: Steps $\langle 5 \rangle 1$ and Lemma 6.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $M.RNext(p) \lor$ UC m.

$\langle 4 \rangle 4$. $R.Reply(p) \lor R.Fail(p)$

$\langle 5 \rangle 1$. $r' \neq r$

PROOF: By propositional logic, it suffices to

ASSUME: $r' = r$

PROVE: FALSE

$\langle 6 \rangle 1$. $\neg M.Return(p)$

PROOF: Steps $\langle 4 \rangle 3$ and $\langle 3 \rangle 1$.

$\langle 6 \rangle 2$. $\neg R.Fail(p)$

PROOF: Assumption $\langle 5 \rangle$ and step $\langle 3 \rangle 2$.

$\langle 6\rangle 3.$ $\neg C.Reply(p)$

PROOF: Steps $\langle 4\rangle 2$ and $\langle 3\rangle 3$.

$\langle 6\rangle 4.$ $h' = h$

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ and the definition of $HNext(p)$ imply $HNext(p) \Rightarrow h' = h$. The assertion follows by assumption 1·1, which implies $HNext(p) \lor (h' = h)$ and simple predicate logic.

$\langle 6\rangle 5.$ Q.E.D.

PROOF: From steps $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 6\rangle 4$, and assumption $\langle 5\rangle$, we conclude UNCHANGED $\langle e, c, r, m, h\rangle$, in contradiction to assumption 1·2.

$\langle 5\rangle 2.$ $R.Next(p)$

PROOF: Step $\langle 5\rangle 1$ and assumption 1·1, which implies $R.Next(p) \lor (r' = r)$.

$\langle 5\rangle 3.$ $\neg R.Forward(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S5$ imply $rstate[p] =$ "B". On the other hand, $R.Forward(p)$ implies $rstate[p] =$ "A".

$\langle 5\rangle 4.$ $\neg R.Reject(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S5$ imply $rstate[p] =$ "B". On the other hand, $R.Reject(p)$ implies $rstate[p] =$ "A".

$\langle 5\rangle 5.$ Q.E.D.

PROOF: Steps $\langle 5\rangle 2$–$\langle 5\rangle 4$ and the definition of $R.Next(p)$.

$\langle 4\rangle 5.$ $S6'$

$\langle 5\rangle 1.$ $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6\rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S5$.

$\langle 6\rangle 2.$ $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6\rangle 3.$ $memCh[p].rbit' = memCh[p].rbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $E.Calling(p)$.

$\langle 5\rangle 2.$ $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S5$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by steps $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5\rangle 3.$ $\mathrm{F} \equiv R.Snd.Calling(p)'$

$\langle 6\rangle 1.$ $crCh[p].cbit' = crCh[p].cbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of c and $C.Rcv.caller(p)$.

$\langle 6\rangle 2.$ $crCh[p].rbit' = crCh[p].cbit$

PROOF: Step $\langle 4\rangle 4$, the definitions of $R.Reply(p)$ and $R.Fail(p)$, which both imply $\exists\, v : R.Snd.Return(p, v)$, and the definition of $R.Snd.Return(p, v)$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Steps ⟨6⟩1 and ⟨6⟩2 imply $crCh[p].rbit' = crCh[p].cbit'$, which by definition equals $\neg R.Snd.Calling(p)'$.

⟨5⟩4. $\text{F} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

PROOF: Tautology.

⟨5⟩5. $\neg\text{F} \wedge (cstate'[p] = \text{“B”})$
$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

⟨6⟩1. CASE: $R.Reply(p)$

⟨7⟩1. $rmCh[p].res \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: Assumption ⟨3⟩ and the definitions of $S5$ and S.

⟨7⟩2. $crCh[p].res' = rmCh[p].res$

PROOF: From the case assumption ⟨6⟩ and the definitions of $R.Reply(p)$ and $R.Snd.Return(p, R.rcvCh[p].res)$.

⟨7⟩3. Q.E.D.

PROOF: Steps ⟨7⟩1 and ⟨7⟩2.

⟨6⟩2. CASE: $R.Fail(p)$

PROOF: From the case assumption and the definitions of $R.Fail(p)$ and $R.Snd.Return(p, \text{“RPCFailure”})$, which imply $crCh[p].res' = \text{“RPCFailure”}$.

⟨6⟩3. Q.E.D.

PROOF: From steps ⟨6⟩1, ⟨6⟩2, and ⟨4⟩4 by propositional logic.

⟨5⟩6. $\text{F} \equiv M.E.Calling(p)'$

⟨6⟩1. $\neg M.E.Calling(p)$

PROOF: Assumption ⟨3⟩ and the definition of $S5$.

⟨6⟩2. $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: By step ⟨4⟩4, because the definitions of $R.Fail(p)$ and $R.Reply(p)$ both imply UC $R.Rcv.caller(p)$.

⟨6⟩3. $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: Step ⟨4⟩3 and the definitions of m and $M.E.rtrner(p)$.

⟨6⟩4. Q.E.D.

PROOF: From steps ⟨6⟩1–⟨6⟩3 by the definition of $M.E.Calling(p)$.

⟨5⟩7. $\text{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology.

⟨5⟩8. $\neg\text{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption ⟨3⟩ and the definitions of $S5$ and S imply $result[p] = NotAResult$. The assertion follows by step ⟨4⟩3, which implies UC $result'[p]$.

⟨5⟩9. $\neg\text{F} \wedge (rstate'[p] = \text{“B”}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: From step ⟨4⟩4, since both $R.Fail(p)$ and $R.Reply(p)$ imply $rstate'[p] = \text{“A”}$.

⟨5⟩10. $\text{“B”} = cstate'[p]$

PROOF: From assumption ⟨3⟩ and the definition of $S5$, which implies $cstate[p] = \text{“B”}$, and step ⟨4⟩2, which implies $cstate'[p] = cstate[p]$.

$\langle 5\rangle 11$. "A" $= rstate'[p]$

Proof: From step $\langle 4\rangle 4$, since both $R.Fail(p)$ and $R.Reply(p)$ imply $rstate'[p] =$ "A".

$\langle 5\rangle 12$. $rmhist'[p] \in$ **if** "B" = "AB" **then** {"A", "B"}
else {"B"}

$\langle 6\rangle 1$. $r' \neq r$

Proof: From step $\langle 4\rangle 4$, we obtain

$crCh[p].rbit' = crCh[p].cbit \neq crCh[p].rbit$

using the definitions of $R.Fail(p)$, $R.Reply(p)$, $R.Snd.Return(p, v)$, and $R.Snd.Calling(p)$. The assertion follows by definition of r, $R.vars(p)$, and $R.Snd.rtrner(p)$.

$\langle 6\rangle 2$. $HNext(p)$

Proof: Step $\langle 6\rangle 1$ and assumption 1·1 (which implies $HNext(p) \lor (r' = r)$).

$\langle 6\rangle 3$. Case: $\neg R.Fail(p) \land R.Reply(p)$

$\langle 7\rangle 1$. $\neg M.Return(p)$

Proof: Steps $\langle 4\rangle 3$ and $\langle 3\rangle 1$.

$\langle 7\rangle 2$. $\neg R.Fail(p)$

Proof: Case assumption $\langle 6\rangle$.

$\langle 7\rangle 3$. $\neg C.Reply(p)$

Proof: Steps $\langle 4\rangle 2$ and $\langle 3\rangle 3$.

$\langle 7\rangle 4$. $rmhist'[p] = rmhist[p]$

Proof: Steps $\langle 7\rangle 1$–$\langle 7\rangle 3$, $\langle 6\rangle 2$, and the definition of $HNext(p)$.

$\langle 7\rangle 5$. Q.E.D.

Proof: Assumption $\langle 3\rangle$ and the definition of $S5$ imply $rmhist[p] =$ "B". The assertion follows by step $\langle 7\rangle 4$.

$\langle 6\rangle 4$. Case: $R.Fail(p)$

Proof: The case assumption, step $\langle 6\rangle 2$, and the definition of $HNext(p)$ imply $rmhist'[p] =$ "B".

$\langle 6\rangle 5$. Q.E.D.

Proof: From steps $\langle 6\rangle 3$, $\langle 6\rangle 4$, and $\langle 4\rangle 4$ by propositional reasoning.

$\langle 5\rangle 13$. $result'[p] \in MemVals \cup \{NotAResult$, "OK", "BadArg"$\}$

Proof: Step $\langle 5\rangle 8$.

$\langle 5\rangle 14$. $\forall\, l \in MemLocs : mem'[l] \in MemVals$

Proof: From assumption $\langle 3\rangle$ and the definitions of $S5$ and S by step $\langle 3\rangle 4$.

$\langle 5\rangle 15$. Q.E.D.

Proof: From steps $\langle 5\rangle 1$–$\langle 5\rangle 14$ by the definitions of $S6'$ and S.

$\langle 4\rangle 6$. Q.E.D.

Proof: Steps $\langle 4\rangle 1$–$\langle 4\rangle 5$.

$\langle 3\rangle 10$. Assume: $S6$

Prove: 1. $C.Reply(p) \lor C.Retry(p)$

2. $C.Reply(p) \Rightarrow S1' \wedge$ UNCHANGED $\langle e, r, m \rangle$
3. $C.Retry(p) \Rightarrow S3' \wedge$ UNCHANGED $\langle e, r, m, h \rangle$

$\langle 4 \rangle 1.$ UNCHANGED e
$\langle 5 \rangle 1.$ $E.Calling(p)$
PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$.
$\langle 5 \rangle 2.$ $\neg E.Next(p)$
PROOF: Steps $\langle 5 \rangle 1$ and Lemma 3.
$\langle 5 \rangle 3.$ Q.E.D.
PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $E.Next(p) \vee$ UC e.
$\langle 4 \rangle 2.$ UNCHANGED r
$\langle 5 \rangle 1.$ $\neg R.Snd.Calling(p)$
PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$.
$\langle 5 \rangle 2.$ $\neg R.Next(p)$
PROOF: Steps $\langle 5 \rangle 1$ and Lemma 5.
$\langle 5 \rangle 3.$ Q.E.D.
PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $R.Next(p) \vee$ UC r.
$\langle 4 \rangle 3.$ UNCHANGED m
$\langle 5 \rangle 1.$ $\neg M.E.Calling(p)$
PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$.
$\langle 5 \rangle 2.$ $\neg M.RNext(p)$
PROOF: Steps $\langle 5 \rangle 1$ and Lemma 6.
$\langle 5 \rangle 3.$ Q.E.D.
PROOF: Step $\langle 5 \rangle 2$ and assumption 1·1, which implies $M.RNext(p) \vee$ UC m.
$\langle 4 \rangle 4.$ $C.Reply(p) \vee C.Retry(p)$
$\langle 5 \rangle 1.$ $c' \neq c$
PROOF: By propositional logic, it suffices to
ASSUME: $c' = c$
PROVE: FALSE
$\langle 6 \rangle 1.$ $\neg M.Return(p)$
PROOF: Steps $\langle 4 \rangle 3$ and $\langle 3 \rangle 1$.
$\langle 6 \rangle 2.$ $\neg R.Fail(p)$
PROOF: Steps $\langle 4 \rangle 2$ and $\langle 3 \rangle 2$.
$\langle 6 \rangle 3.$ $\neg C.Reply(p)$
PROOF: Assumption $\langle 5 \rangle$ and step $\langle 3 \rangle 3$.
$\langle 6 \rangle 4.$ $h' = h$
PROOF: Steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ and the definition of $HNext(p)$ imply $HNext(p) \Rightarrow h' = h$. The assertion follows by assumption 1·1, which implies $HNext(p) \vee (h' = h)$, and simple predicate logic.
$\langle 6 \rangle 5.$ Q.E.D.
PROOF: From steps $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$, $\langle 6 \rangle 4$, and assumption $\langle 5 \rangle$, we conclude

UNCHANGED $\langle e, c, r, m, h \rangle$, in contradiction to assumption 1·2.

$\langle 5 \rangle 2$. $C.Next(p)$

PROOF: Step $\langle 5 \rangle 1$ and assumption 1·1, which implies $C.Next(p) \lor (c' = c)$.

$\langle 5 \rangle 3$. $\neg C.Forward(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$ implies $cstate[p] =$ "B". On the other hand, $C.Forward(p)$ implies $cstate[p] =$ "A".

$\langle 5 \rangle 4$. Q.E.D.

PROOF: Steps $\langle 5 \rangle 2$ and $\langle 5 \rangle 3$ and the definition of $C.Next(p)$.

$\langle 4 \rangle 5$. ASSUME: $C.Reply(p)$
PROVE: $S1'$

$\langle 5 \rangle 1$. $\mathrm{F} \equiv E.Calling(p)'$

$\langle 6 \rangle 1$. $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4 \rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6 \rangle 2$. $memCh[p].rbit' = memCh[p].cbit$

PROOF: From assumption $\langle 4 \rangle$, using the definitions of $C.Reply(p)$ and $C.Snd.Return(p, v)$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$ imply $memCh[p].cbit' = memCh[p].rbit'$, which by definition equals $\neg E.Calling(p)'$.

$\langle 5 \rangle 2$. $\mathrm{F} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Tautology.

$\langle 5 \rangle 3$. $\mathrm{F} \equiv R.Snd.Calling(p)'$

$\langle 6 \rangle 1$. $\neg R.Snd.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$.

$\langle 6 \rangle 2$. $crCh[p].cbit' = crCh[p].cbit$

PROOF: Assumption $\langle 4 \rangle$ and the definition of $C.Reply(p)$, which implies UC $C.Rcv.caller(p)$, and the definition of $C.Rcv.caller(p)$.

$\langle 6 \rangle 3$. $crCh[p].rbit' = crCh[p].rbit$

PROOF: Step $\langle 4 \rangle 2$ and the definitions of r and $R.Snd.rtrner(p)$.

$\langle 6 \rangle 4$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$–$\langle 6 \rangle 3$ and the definition of $\neg R.Snd.Calling(p)$.

$\langle 5 \rangle 4$. $\mathrm{F} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

PROOF: Tautology.

$\langle 5 \rangle 5$. $\neg \mathrm{F} \land (cstate'[p] = \text{"B"})$
$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}, \text{"RPCFailure"}\}$

PROOF: From assumption $\langle 4 \rangle$ and the definition of $C.Reply(p)$, which implies $cstate'[p] =$ "A".

$\langle 5 \rangle 6$. $\mathrm{F} \equiv M.E.Calling(p)'$

$\langle 6 \rangle 1$. $\neg M.E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle$ and the definition of $S6$.

$\langle 6\rangle 2.$ $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of r and $R.Rcv.caller(p)$.

$\langle 6\rangle 3.$ $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: Step $\langle 4\rangle 3$ and the definitions of m and $M.E.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ by the definition of $M.E.Calling(p)$.

$\langle 5\rangle 7.$ $\mathrm{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology.

$\langle 5\rangle 8.$ $\neg\mathrm{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S6$ and S imply $result[p] = NotAResult$. The assertion follows by step $\langle 4\rangle 3$, which implies UC $result'[p]$.

$\langle 5\rangle 9.$ $\neg\mathrm{F} \wedge (rstate'[p] = \text{"B"}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}\}$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S6$ imply $rstate[p] = \text{"A"}$. By step $\langle 4\rangle 2$, we conclude $rstate'[p] = \text{"A"}$.

$\langle 5\rangle 10.$ $\text{"A"} = cstate'[p]$

PROOF: Assumption $\langle 4\rangle$ and the definition of $C.Reply(p)$.

$\langle 5\rangle 11.$ $\text{"A"} = rstate'[p]$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S6$ imply $rstate[p] = \text{"A"}$. The assertion follows by step $\langle 4\rangle 2$, which implies UC $rstate[p]$.

$\langle 5\rangle 12.$ $rmhist'[p] \in \textbf{if } \text{"A"} = \text{"AB"} \textbf{ then } \{\text{"A"}, \text{"B"}\} \textbf{ else } \{\text{"A"}\}$

$\langle 6\rangle 1.$ $c' \neq c$

PROOF: Assumption $\langle 4\rangle$ and the definition of $C.Reply(p)$ imply $cstate'[p] \neq cstate[p]$. The assertion follows by definition of c.

$\langle 6\rangle 2.$ $HNext(p)$

PROOF: Step $\langle 6\rangle 1$ and assumption 1·1, which implies $HNext(p) \vee (c' = c)$.

$\langle 6\rangle 3.$ $\neg M.Return(p)$

PROOF: Steps $\langle 4\rangle 3$ and $\langle 3\rangle 1$.

$\langle 6\rangle 4.$ $\neg R.Fail(p)$

PROOF: Steps $\langle 4\rangle 2$ and $\langle 3\rangle 2$.

$\langle 6\rangle 5.$ Q.E.D.

PROOF: By steps $\langle 6\rangle 2$–$\langle 6\rangle 4$ by the definition of $HNext(p)$.

$\langle 5\rangle 13.$ $result'[p] \in MemVals \cup \{NotAResult, \text{"OK"}, \text{"BadArg"}\}$

PROOF: Step $\langle 5\rangle 8$.

$\langle 5\rangle 14.$ $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: From assumption $\langle 3\rangle$ and the definitions of $S6$ and S by step $\langle 3\rangle 4$.

$\langle 5\rangle 15.$ Q.E.D.

PROOF: From steps $\langle 5\rangle 1$–$\langle 5\rangle 14$ by the definitions of $S1'$ and S'.

$\langle 4\rangle 6.$ ASSUME: $C.Retry(p)$

PROVE: UNCHANGED h

$\langle 5\rangle 1.$ $\neg M.Return(p)$

PROOF: Steps $\langle 4\rangle 3$ and $\langle 3\rangle 1$.

$\langle 5\rangle 2.$ $\neg R.Fail(p)$

PROOF: Steps $\langle 4\rangle 2$ and $\langle 3\rangle 2$.

$\langle 5\rangle 3.$ $\neg C.Reply(p)$

PROOF: Assumption $\langle 4\rangle$ and the definition of $C.Retry(p)$ imply UC $cstate[p]$. On the other hand, $C.Reply(p)$ implies $cstate'[p] \neq cstate[p]$.

$\langle 5\rangle 4.$ $HNext(p) \Rightarrow h' = h$

PROOF: From steps $\langle 5\rangle 1$–$\langle 5\rangle 3$ by the definition of $HNext(p)$.

$\langle 5\rangle 5.$ Q.E.D.

PROOF: From step $\langle 5\rangle 4$ and assumption 1·1, which implies $HNext(p) \vee (h' = h)$, by simple predicate logic.

$\langle 4\rangle 7.$ ASSUME: $C.Retry(p)$

PROVE: $S3'$

$\langle 5\rangle 1.$ $\mathrm{T} \equiv E.Calling(p)'$

$\langle 6\rangle 1.$ $.Calling(p)$

PROOF: Assumption $\langle 3\rangle$ and the definition of $S6$.

$\langle 6\rangle 2.$ $memCh[p].cbit' = memCh[p].cbit$

PROOF: Step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 6\rangle 3.$ $memCh[p].rbit' = memCh[p].rbit$

PROOF: From assumption $\langle 4\rangle$, using the definitions of $C.Retry(p)$ and $C.Snd.rtrner(p)$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: From steps $\langle 6\rangle 1$–$\langle 6\rangle 3$ and the definition of $E.Calling(p)$.

$\langle 5\rangle 2.$ $\mathrm{T} \Rightarrow (memCh[p].arg' \in LegalArgs)$

PROOF: Assumption $\langle 3\rangle$ and the definitions of $S6$ and S imply $memCh[p].arg \in LegalArgs$. The assertion follows by step $\langle 4\rangle 1$ and the definitions of e and $E.caller(p)$.

$\langle 5\rangle 3.$ $\mathrm{T} \equiv R.Snd.Calling(p)'$

$\langle 6\rangle 1.$ $crCh[p].cbit' \neq crCh[p].rbit$

PROOF: From assumption $\langle 4\rangle$, using the definitions of $C.Retry(p)$ and $C.Rcv.Call(p, v)$.

$\langle 6\rangle 2.$ $crCh[p].rbit' = crCh[p].rbit$

PROOF: Step $\langle 4\rangle 2$ and the definitions of r and $R.Snd.rtrner(p)$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$ imply $crCh[p].cbit' \neq crCh[p].rbit$, which by definition equals $R.Snd.Calling(p)'$.

$\langle 5\rangle 4.$ $\mathrm{T} \Rightarrow (crCh[p].arg' = C.RelayArg(p)')$

$\langle 6\rangle 1.$ $crCh[p].arg' = C.RelayArg(p)$

PROOF: From assumption $\langle 4\rangle$, using the definitions of $C.Retry(p)$ and

$C.Rcv.Call(p, C.RelayArg(p))$.

⟨6⟩2. $C.RelayArg(p)' = C.RelayArg(p)$

PROOF: Step ⟨4⟩1 implies UC $memCh[p].arg$, using the definitions of e and $E.caller(p)$. The assertion follows by the definition of $C.RelayArg(p)$.

⟨6⟩3. Q.E.D.

PROOF: Steps ⟨6⟩1 and ⟨6⟩2.

⟨5⟩5. $\neg \mathrm{T} \wedge (cstate'[p] = \text{“B”})$

$\Rightarrow crCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}, \text{“RPCFailure”}\}$

PROOF: Tautology.

⟨5⟩6. $\mathrm{F} \equiv M.E.Calling(p)'$

⟨6⟩1. $\neg M.E.Calling(p)$

PROOF: Assumption ⟨3⟩ and the definition of $S6$.

⟨6⟩2. $rmCh[p].cbit' = rmCh[p].cbit$

PROOF: Step ⟨4⟩2 and the definitions of r and $R.Rcv.caller(p)$.

⟨6⟩3. $rmCh[p].rbit' = rmCh[p].rbit$

PROOF: Step ⟨4⟩3 and the definitions of m and $M.E.rtrner(p)$.

⟨6⟩4. Q.E.D.

PROOF: From steps ⟨6⟩1–⟨6⟩3 by the definition of $M.E.Calling(p)$.

⟨5⟩7. $\mathrm{F} \Rightarrow (rmCh[p].arg' = R.RelayArg(p)')$

PROOF: Tautology.

⟨5⟩8. $\neg \mathrm{F} \Rightarrow (result'[p] = NotAResult)$

PROOF: Assumption ⟨3⟩ and the definitions of $S6$ and S imply $result[p] = NotAResult$. The assertion follows by step ⟨4⟩3, which implies UC $result[p]$.

⟨5⟩9. $\neg \mathrm{F} \wedge (rstate'[p] = \text{“B”}) \Rightarrow rmCh[p].res' \in MemVals \cup \{\text{“OK”}, \text{“BadArg”}\}$

PROOF: From assumption ⟨3⟩ and the definition of $S6$, which implies $rstate[p] =$ “A”, and step ⟨4⟩2, which implies UC $rstate[p]$.

⟨5⟩10. $\text{“B”} = cstate'[p]$

PROOF: From assumption ⟨4⟩ and the definition of $C.Retry(p)$, which implies $cstate[p] = \text{“B”}$ and UC $cstate[p]$.

⟨5⟩11. $\text{“A”} = rstate'[p]$

PROOF: Assumption ⟨3⟩ and the definition of $S6$ imply $rstate[p] = \text{“A”}$. The assertion follows by step ⟨4⟩2, which implies UC $rstate[p]$.

⟨5⟩12. $rmhist'[p] \in \textbf{if } \text{“AB”} = \text{“AB”} \textbf{ then } \{\text{“A”}, \text{“B”}\} \textbf{ else } \{\text{“AB”}\}$

⟨6⟩1. $rmhist[p] = \text{“B”}$

PROOF: Assumption ⟨3⟩ and the definition of $S6$.

⟨6⟩2. $HNext(p) \Rightarrow (rmhist'[p] \in \{rmhist[p], \text{“A”}, \text{“B”}\})$

PROOF: Immediate from the definition of $HNext(p)$.

⟨6⟩3. Q.E.D.

PROOF: By predicate logic from steps ⟨6⟩1 and ⟨6⟩2 and assumption 1·1, which implies $HNext(p) \lor \textsc{uc}\ rmhist[p]$.

$\langle 5 \rangle 13$. $result'[p] \in MemVals \cup \{NotAResult, \text{“OK”}, \text{“BadArg”}\}$

PROOF: Step $\langle 5 \rangle 8$.

$\langle 5 \rangle 14$. $\forall\, l \in MemLocs : mem'[l] \in MemVals$

PROOF: From assumption $\langle 3 \rangle$ and the definitions of $S6$ and S by step $\langle 3 \rangle 4$.

$\langle 5 \rangle 15$. Q.E.D.

PROOF: From steps $\langle 5 \rangle 1$–$\langle 5 \rangle 14$ by the definitions of $S3'$ and S'.

$\langle 4 \rangle 8$. Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$–$\langle 4 \rangle 7$ by propositional reasoning.

$\langle 3 \rangle 11$. Q.E.D.

PROOF: From steps $\langle 3 \rangle 5$–$\langle 3 \rangle 10$ by first-order logic.

1.3. $S1 \Rightarrow \overline{PInit(p)}$

By propositional reasoning, it suffices to

ASSUME: $S1$

PROVE: $\overline{PInit}(p)$

$\langle 3 \rangle 1$. $\overline{result}[p] = result[p]$

PROOF: Immediate from assumption 1 and the definition of $\overline{result}[p]$.

$\langle 3 \rangle 2$. $result[p] = NotAResult$

PROOF: Assumption 1 and the definition of $S1$ imply $\neg M.E.Calling$, which in turn implies the assertion, by the definition of S.

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

PROOF: From steps $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$ by the definition of $\overline{PInit(p)}$.

1.4. 1. $S1 \land S2' \land E.Next(p) \land \textsc{uc}\ \langle c, r, m \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

2. $S2 \land S3' \land C.Forward(p) \land \textsc{uc}\ \langle e, r, m, h \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

3. a. $S3 \land S4' \land R.Forward(p) \land \textsc{uc}\ \langle e, c, m, h \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

b. $S3 \land S6' \land R.Fail(p) \land \textsc{uc}\ \langle e, c, m \rangle \Rightarrow \overline{Fail(p)}$

4. a. $S4 \land S4' \land M.Read(p) \land \textsc{uc}\ \langle e, c, r, h \rangle \Rightarrow \overline{Read(p)}$

b. ASSUME: l : CONSTANT

PROVE: $S4 \land S4' \land M.Write(p, l) \land \textsc{uc}\ \langle e, c, r, h \rangle \Rightarrow \overline{Write(p, l)}$

c. $S4 \land S5' \land M.Return(p) \land \textsc{uc}\ \langle e, c, r \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

5. a. $S5 \land S6' \land R.Reply(p) \land \textsc{uc}\ \langle e, c, m \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

b. $S5 \land S6' \land R.Fail(p) \land \textsc{uc}\ \langle e, c, m \rangle \Rightarrow \overline{Fail(p)}$

6. a. $S6 \land S1' \land C.Reply(p) \land \textsc{uc}\ \langle e, r, m \rangle \Rightarrow \overline{Return(p)}$

b. $S6 \land S3' \land C.Retry(p) \land \textsc{uc}\ \langle e, r, m, h \rangle \Rightarrow \overline{Fail(p)}$

7. $\textsc{uc}\ \langle e, c, r, m, h \rangle \Rightarrow \textsc{uc}\ \overline{pv}$

$\langle 3 \rangle 1$. ASSUME: 1. $S1$

2. $S2'$

3. $E.Next(p)$

4. UNCHANGED $\langle c, r, m \rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4\rangle 1$. UNCHANGED $E.rtrner(p)$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of c, since $C.Snd.rtrner(p) = E.rtrner(p)$.

$\langle 4\rangle 2$. $\overline{result}[p] = result[p]$

PROOF: Assumption $\langle 3\rangle{\cdot}1$ and the definition of $\overline{result}[p]$.

$\langle 4\rangle 3$. $\overline{result'}[p] = result'[p]$

PROOF: Assumption $\langle 3\rangle{\cdot}2$ and the definition of $\overline{result'}[p]$.

$\langle 4\rangle 4$. UNCHANGED $result[p]$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of m.

$\langle 4\rangle 5$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$–$\langle 4\rangle 4$ by the definition of $\overline{pv}$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3\rangle 2$. ASSUME: 1. $S2$
2. $S3'$
3. $C.Forward(p)$
4. UNCHANGED $\langle e, r, m, h\rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4\rangle 1$. UNCHANGED $E.rtrner(p)$

PROOF: From assumption $\langle 3\rangle 3$, using the definition of $C.Forward(p)$, since $C.Snd.rtrner(p) = E.rtrner(p)$.

$\langle 4\rangle 2$. $\overline{result}[p] = result[p]$

PROOF: Assumption $\langle 3\rangle 1$ and the definition of $\overline{result}[p]$.

$\langle 4\rangle 3$. $\overline{result'}[p] = result'[p]$

$\langle 5\rangle 1$. $h = $ "A"

PROOF: Assumption $\langle 3\rangle 1$ and the definition of $S2$.

$\langle 5\rangle 2$. $h' = $ "A"

PROOF: Assumption $\langle 3\rangle 4$ and step $\langle 5\rangle 1$.

$\langle 5\rangle 3$. Q.E.D.

PROOF: From assumption $\langle 3\rangle{\cdot}2$ and step $\langle 5\rangle 2$ by the definition of $\overline{result'}[p]$.

$\langle 4\rangle 4$. UNCHANGED $result[p]$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of m.

$\langle 4\rangle 5$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$–$\langle 4\rangle 4$ by the definition of $\overline{pv}$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3\rangle 3$. ASSUME: 1. $S3$
2. $S4'$
3. $R.Forward(p)$
4. UNCHANGED $\langle e, c, m, h\rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4\rangle 1$. UNCHANGED $E.rtrner(p)$

PROOF: By assumption $\langle 3\rangle 4$ and the definition of c, since $C.Snd.rtrner(p) = E.rtrner(p)$.

$\langle 4\rangle 2$. UNCHANGED $\overline{result}[p]$

$\langle 5\rangle 1$. CASE: $h =$ "A"

$\langle 6\rangle 1$. $\overline{result}[p] = result[p]$

PROOF: Assumption $\langle 3\rangle 1$, case assumption $\langle 5\rangle$, and the definition of $\overline{result}[p]$.

$\langle 6\rangle 2$. $h' =$ "A"

PROOF: Case assumption $\langle 5\rangle$ and assumption $\langle 3\rangle 4$.

$\langle 6\rangle 3$. $\overline{result'}[p] = result'[p]$

PROOF: From assumption $\langle 3\rangle{\cdot}2$ and step $\langle 6\rangle 2$ by the definition of $\overline{result'}[p]$.

$\langle 6\rangle 4$. UNCHANGED $result[p]$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of m.

$\langle 6\rangle 5$. Q.E.D.

PROOF: Steps $\langle 6\rangle 1$, $\langle 6\rangle 3$, and $\langle 6\rangle 4$.

$\langle 5\rangle 2$. CASE: $h =$ "B"

$\langle 6\rangle 1$. $\overline{result}[p] =$ "MemFailure"

PROOF: From assumption $\langle 3\rangle{\cdot}1$ and the case assumption $\langle 5\rangle$ by the definition of $\overline{result}[p]$.

$\langle 6\rangle 2$. $h' =$ "B"

PROOF: Case assumption $\langle 5\rangle$ and assumption $\langle 3\rangle{\cdot}4$.

$\langle 6\rangle 3$. $result[p] = NotAResult$

PROOF: From assumption $\langle 3\rangle{\cdot}1$ and the definition of $S3$, which implies $\neg M.E.Calling(p)$, by the definition of S.

$\langle 6\rangle 4$. $result'[p] = NotAResult$

PROOF: Step $\langle 6\rangle 3$ and assumption $\langle 3\rangle{\cdot}4$, which implies UNCHANGED $result[p]$.

$\langle 6\rangle 5$. $\overline{result'}[p] =$ "MemFailure"

PROOF: From steps $\langle 6\rangle 2$ and $\langle 6\rangle 4$ by assumption $\langle 3\rangle{\cdot}2$, lemma 11, and the definition of $\overline{result'}[p]$.

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

PROOF: Steps $\langle 6\rangle 1$ and $\langle 6\rangle 5$.

$\langle 5\rangle 3$. Q.E.D.

PROOF: From steps $\langle 5\rangle 1$ and $\langle 5\rangle 2$ by assumption $\langle 3\rangle{\cdot}1$, which (by the definition of $S3$) implies $h \in \{$"A", "B"$\}$.

$\langle 4\rangle 3$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$ and $\langle 4\rangle 2$ by the definition of $\overline{pv}$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3\rangle 4$. ASSUME: 1. $S3$
2. $S6'$
3. $R.Fail(p)$
4. UNCHANGED $\langle e, c, m\rangle$

PROVE: $\overline{Fail(p)}$

$\langle 4\rangle 1$. $E.calling(p)$

PROOF: Assumption $\langle 3\rangle{\cdot}1$ and the definition of $S3$.

$\langle 4\rangle 2$. $crCh[p].res' =$ "RPCFailure"

PROOF: From assumption $\langle 3\rangle 3$, expanding the definitions of $R.Fail(p)$ and $R.Snd.Return(p,$ "RPCFailure"$)$.

$\langle 4\rangle 3$. $\overline{result'}[p] =$ "MemFailure"

PROOF: From assumption $\langle 3\rangle{\cdot}2$, lemma 14, and step $\langle 4\rangle 2$ by the definition of $\overline{result'}[p]$.

$\langle 4\rangle 4$. UNCHANGED $E.rtrner(p)$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of c, since $C.Snd.rtrner(p)$ is $E.rtrner(p)$.

$\langle 4\rangle 5$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$, $\langle 4\rangle 2$, and $\langle 4\rangle 4$ by the definition of $\overline{Fail(p)}$, since $E.Calling(p) \equiv \overline{E.Calling(p)}$ and $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3\rangle 5$. ASSUME: 1. l : CONSTANT

2. $S4$

3. $S4'$

4. $M.Read(p, l)$

5. UNCHANGED $\langle e, c, r, h\rangle$

PROVE: $\overline{Read(p, l)}$

$\langle 4\rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3\rangle{\cdot}2$ and the definition of $S4$.

$\langle 4\rangle 2$. $memCh[p].arg = \langle$ "Read", $l\rangle$

$\langle 5\rangle 1$. $rmCh[p].arg = \langle$ "Read", $l\rangle >$

PROOF: Assumption $\langle 3\rangle{\cdot}4$ and the definition of $M.Read(p, l)$.

$\langle 5\rangle 2$. $rmCh[p].arg = R.RelayArg(p)$

PROOF: Assumption $\langle 3\rangle{\cdot}2$ and the definitions of $S4$ and S.

$\langle 5\rangle 3$. Q.E.D.

PROOF: From steps $\langle 5\rangle 1$ and $\langle 5\rangle 2$ by lemma 7, which is applicable by assumption $\langle 3\rangle{\cdot}2$ and the definition of $S4$.

$\langle 4\rangle 3$. $\overline{result'}[p] =$ **if** $l \in MemLocs$ **then** $mem[l]$ **else** "BadArg"

$\langle 5\rangle 1$. $result'[p] \neq NotAResult$

$\langle 6\rangle 1$. CASE: $l \in MemLocs$

PROOF: From assumption $\langle 3\rangle{\cdot}4$, the case assumption, and the definition of $M.Read(p, l)$, we obtain $result'[p] = mem[l]$, which in turn implies $result'[p] \in MemVals$, using assumption $\langle 3\rangle{\cdot}3$ and the definitions of $S4'$ and S'. The assertion follows by the definition of $NotAResult$, which implies $NotAResult \notin MemVals$ (using the axioms of set theory, which ensure the existence of some

$x \notin S$, for any set S).

$\langle 6 \rangle 2$. CASE: $\neg(l \in \mathit{MemLocs})$

PROOF: From assumption $\langle 3 \rangle{\cdot}4$, the case assumption, and the definition of $M.\mathit{Read}(p, l)$, we obtain $\mathit{result}'[p] =$ "BadArg". The assertion follows by a similar argument as in the proof of step $\langle 6 \rangle 1$.

$\langle 6 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 6 \rangle 1$ and $\langle 6 \rangle 2$.

$\langle 5 \rangle 2$. $\overline{\mathit{result}'}[p] = \mathit{result}'[p]$

PROOF: Assumption $\langle 3 \rangle{\cdot}3$, step $\langle 5 \rangle 1$, lemma 12, and the definition of $\overline{\mathit{result}'}[p]$.

$\langle 5 \rangle 3$. Q.E.D.

PROOF: Step $\langle 5 \rangle 2$, assumption $\langle 3 \rangle{\cdot}4$, and the definition of $M.\mathit{Read}(p, l)$.

$\langle 4 \rangle 4$. UNCHANGED $E.\mathit{rtrner}(p)$

PROOF: From assumption $\langle 3 \rangle{\cdot}5$ and the definition of c, since $E.\mathit{rtrner}(p)$ is $C.\mathit{Snd}.\mathit{rtrner}(p)$.

$\langle 4 \rangle 5$. Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$–$\langle 4 \rangle 4$ by the definition of $\overline{\mathit{Read}(p)}$, since $\overline{E.\mathit{Calling}(p)} \equiv \overline{E.\mathit{Calling}(p)}$, $\overline{\mathit{memCh}[p].\mathit{arg}} = \mathit{memCh}[p].\mathit{arg}$, and $\overline{E.\mathit{rtrner}(p)} = E.\mathit{rtrner}(p)$.

$\langle 3 \rangle 6$. ASSUME: 1. l, v : CONSTANT

2. $S4$

3. $S4'$

4. $M.\mathit{Write}(p, l, v)$

5. UNCHANGED $\langle e, c, r, h \rangle$

PROVE: $\overline{\mathit{Write}(p, l, v)}$

PROOF: Similar to the proof of step $\langle 3 \rangle 6$, using the definition of $M.\mathit{Write}(p, l, v)$.

$\langle 3 \rangle 7$. ASSUME: 1. $S4$

2. $S5'$

3. $M.\mathit{Return}(p)$

4. UNCHANGED $\langle e, c, r \rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4 \rangle 1$. UNCHANGED $E.\mathit{rtrner}(p)$

PROOF: From assumption $\langle 3 \rangle{\cdot}4$ and the definition of c, since $E.\mathit{rtrner}(p)$ is $C.\mathit{Snd}.\mathit{rtrner}(p)$.

$\langle 4 \rangle 2$. $\overline{\mathit{result}}[p] = \mathit{result}[p]$

PROOF: Assumption $\langle 3 \rangle{\cdot}3$ and the definition of $M.\mathit{Return}(p)$ imply $\mathit{result}[p] \neq \mathit{NotAResult}$. The assertion follows by the definition of $\overline{\mathit{result}}[p]$, using assumption $\langle 3 \rangle{\cdot}1$ and lemma 12.

$\langle 4 \rangle 3$. $\overline{\mathit{result}'}[p] = \mathit{rmCh}[p].\mathit{res}'$

PROOF: By assumption $\langle 3 \rangle{\cdot}2$, lemma 13, and the definition of $\overline{\mathit{result}'}[p]$.

$\langle 4 \rangle 4$. $\mathit{rmCh}[p].\mathit{res}' = \mathit{result}[p]$

PROOF: From assumption $\langle 3 \rangle{\cdot}3$, using the definitions of $M.\mathit{Return}(p)$ and

$M.Snd.Return(p, result[p])$.

$\langle 4 \rangle 5$. Q.E.D.

PROOF: Steps $\langle 4 \rangle 1$–$\langle 4 \rangle 4$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3 \rangle 8$. ASSUME: 1. $S5$

2. $S6'$

3. $R.Reply(p)$

4. UNCHANGED $\langle e, c, m \rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4 \rangle 1$. UNCHANGED $E.rtrner(p)$

PROOF: From assumption $\langle 3 \rangle \cdot 4$, using the definition of c, since $E.rtrner(p)$ is $C.Snd.rtrner(p)$.

$\langle 4 \rangle 2$. $\overline{result}[p] = rmCh[p].res$

PROOF: From assumption $\langle 3 \rangle \cdot 1$ and the definition of $\overline{result}[p]$ by lemma 13.

$\langle 4 \rangle 3$. $crCh[p].res' = rmCh[p].res$

PROOF: From assumption $\langle 3 \rangle \cdot 3$, expanding the definitions of $R.Reply(p)$ and $R.Snd.Return(p, R.rcvCh[p].res)$.

$\langle 4 \rangle 4$. $rmCh[p].res \neq$ "RPCFailure"

PROOF: By assumption $\langle 3 \rangle \cdot 1$ and the definitions of $S5$ and S, we obtain $rmCh[p].res \in MemVals \cup \{$"OK", "BadArg"$\}$. Assumption $FailureNotAValue$ asserts $RPCFailure \notin MemVals$ and therefore implies the assertion.

$\langle 4 \rangle 5$. $\overline{result'}[p] = crCh[p].res'$

PROOF: From steps $\langle 4 \rangle 3$ and $\langle 4 \rangle 4$ by the definition of $\overline{result'}[p]$ and lemma 14.

$\langle 4 \rangle 6$. Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$, and $\langle 4 \rangle 5$ by the definition of $\overline{pv}$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3 \rangle 9$. ASSUME: 1. $S5$

2. $S6'$

3. $R.Fail(p)$

4. UNCHANGED $\langle e, c, m \rangle$

PROVE: $\overline{Fail(p)}$

$\langle 4 \rangle 1$. $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle \cdot 1$ and the definition of $S5$.

$\langle 4 \rangle 2$. $\overline{result'}[p] =$ "MemFailure"

PROOF: Assumption $\langle 3 \rangle \cdot 3$ implies $crCh[p].res' =$ "RPCFailure", using the definitions of $R.Fail(p)$ and $R.Snd.Return(p,$ "RPCFailure"$)$. The assertion follows by assumption $\langle 3 \rangle \cdot 2$ and lemma 14 from the definition of $\overline{result'}[p]$.

$\langle 4 \rangle 3$. UNCHANGED $E.rtrner(p)$

PROOF: From assumption $\langle 3 \rangle \cdot 4$, using the definition of c, since $E.rtrner(p)$ is $C.Snd.rtrner(p)$.

$\langle 4 \rangle 4$. Q.E.D.

PROOF: From steps $\langle 4\rangle 1$–$\langle 4\rangle 3$ and the definition of $\overline{Fail(p)}$, since $\overline{E.Calling(p)} \equiv E.Calling(p)$ and $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3\rangle 10$. ASSUME: 1. $S6$
2. $S1'$
3. $C.Reply(p)$
4. UNCHANGED $\langle e, r, m \rangle$
PROVE: $\overline{Return(p)}$

$\langle 4\rangle 1$. $\overline{result}[p] = C.ReplyVal(p)$
PROOF: From assumption $\langle 3\rangle{\cdot}1$ by lemma 14 and the definitions of $\overline{result}[p]$ and $C.ReplyVal(p)$, since $C.rcvCh$ is $crCh$.

$\langle 4\rangle 2$. $\overline{result}[p] \neq NotAResult$

$\langle 5\rangle 1$. CASE: $crCh[p].res =$ "RPCFailure"
PROOF: The case assumption, assumption $\langle 3\rangle{\cdot}1$, and lemma 14 imply $\overline{result}[p] =$ "MemFailure", from which the assertion follows by the definition of $NotAResult$, using the axioms of set theory, which imply the existence of some value $x \notin S$, for any set S.

$\langle 5\rangle 2$. CASE: $crCh[p].res \neq$ "RPCFailure"
PROOF: The case assumption, assumption $\langle 3\rangle{\cdot}1$, and lemma 14 imply $\overline{result}[p] = crCh[p].res$. Assumption $\langle 3\rangle{\cdot}1$, the case assumption, and the definitions of $S6$ and S imply

$$crCh[p].res \in MemVals \cup \{\text{"OK"}, \text{"BadArg"}\}$$

from which the assertion follows by the definition of $NotAResult$, using a similar argument as in the proof of step $\langle 5\rangle 1$.

$\langle 5\rangle 3$. Q.E.D.
PROOF: Steps $\langle 5\rangle 1$ and $\langle 5\rangle 2$.

$\langle 4\rangle 3$. $\overline{result'}[p] = NotAResult$
PROOF: Assumption $\langle 3\rangle{\cdot}2$ and the definition of $\overline{result'}[p]$ imply $\overline{result'}[p] = result'[p]$. Now, assumption $\langle 3\rangle{\cdot}2$ and the definitions of $S1'$ and S' imply $result'[p] = NotAResult$.

$\langle 4\rangle 4$. $E.Return(p, \overline{result}[p])$
PROOF: Assumption $\langle 3\rangle{\cdot}3$, step $\langle 4\rangle 1$, and the definitions of $C.Reply(p)$ and $C.Snd.Return(p, C.ReplyVal(p))$.

$\langle 4\rangle 5$. Q.E.D.
PROOF: By the definition of $\overline{pv}$ from steps $\langle 4\rangle 2$–$\langle 4\rangle 4$, since

$$\overline{E.Return(p, result[p])} \equiv E.Return(p, \overline{result}[p])$$

$\langle 3\rangle 11$. ASSUME: 1. $S6$
2. $S3'$
3. $C.Retry(p)$
4. UNCHANGED $\langle e, r, m, h \rangle$
PROVE: $\overline{Fail(p)}$

$\langle 4 \rangle 1.$ $E.Calling(p)$

PROOF: Assumption $\langle 3 \rangle{\cdot}1$ and the definition of $S6$.

$\langle 4 \rangle 2.$ $\overline{result'}[p] =$ "MemFailure"

$\langle 5 \rangle 1.$ $h =$ "B"

PROOF: Assumption $\langle 3 \rangle{\cdot}1$ and the definition of $S6$.

$\langle 5 \rangle 2.$ $h' =$ "B"

PROOF: Step $\langle 5 \rangle 1$ and assumption $\langle 3 \rangle{\cdot}4$.

$\langle 5 \rangle 3.$ Q.E.D.

PROOF: Assumption $\langle 3 \rangle{\cdot}2$, step $\langle 5 \rangle 2$, the definition of $\overline{result'}[p]$, and lemma 11.

$\langle 4 \rangle 3.$ UNCHANGED $E.rtrner(p)$

PROOF: Assumption $\langle 3 \rangle{\cdot}3$ and the definition of $C.Retry(p)$, observing that $C.Snd.rtrner(p)$ is $E.rtrner(p)$.

$\langle 4 \rangle 4.$ Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$–$\langle 4 \rangle 3$ by the definition of $\overline{Fail(p)}$, since $\overline{E.Calling(p)} \equiv E.Calling(p)$ and $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3 \rangle 12.$ ASSUME: 1. $S1 \lor S2 \lor S3 \lor S4 \lor S5 \lor S6$

2. UNCHANGED $\langle e, c, r, m, h \rangle$

PROVE: UNCHANGED $\overline{pv}$

$\langle 4 \rangle 1.$ UNCHANGED $E.rtrner(p)$

PROOF: Assumption $\langle 3 \rangle{\cdot}2$ and the definition of c, since $C.Snd.rtrner(p)$ is $E.rtrner(p)$.

$\langle 4 \rangle 2.$ UNCHANGED $\overline{result}[p]$

$\langle 5 \rangle 1.$ $S1 \equiv S1'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 2.$ $S2 \equiv S2'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 3.$ $S3 \equiv S3'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 4.$ $S4 \equiv S4'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 5.$ $S5 \equiv S5'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 6.$ $S6 \equiv S6'$

PROOF: From assumption $\langle 3 \rangle{\cdot}2$ by lemma 8.

$\langle 5 \rangle 7.$ Q.E.D.

PROOF: From assumption $\langle 3 \rangle{\cdot}1$ and steps $\langle 5 \rangle 1$–$\langle 5 \rangle 6$ by the definition of $\overline{result}[p]$, all of whose free variables appear in $\langle e, c, r, m, h \rangle$.

$\langle 4 \rangle 3.$ Q.E.D.

PROOF: From steps $\langle 4 \rangle 1$ and $\langle 4 \rangle 2$ by the definition of $\overline{pv}$, since $\overline{E.rtrner(p)} = E.rtrner(p)$.

$\langle 3 \rangle 13$. Q.E.D.

PROOF: From steps $\langle 3 \rangle 1$–$\langle 3 \rangle 12$ by predicate logic.

1.5. Q.E.D.

3.2 The Proof of Step 1.5

We now give the high-level proof of step 1.5, which is the only part of the proof of step 1 that involves temporal logic.

LET: $Inv(p) \triangleq S1 \lor S2 \lor S3 \lor S4 \lor S5 \lor S6$

1.5.1. $IPImp(p) \Rightarrow \Box Inv(p)$

PROOF: By steps 1.1, 1.2, and the laws of TLA, which allow us in general to deduce $P \land (\forall\, u \in U : \Box[N(u)]_{v(u)}) \Rightarrow \Box I$ from $P \Rightarrow I$ together with $I \land (\forall\, u \in U : [N(u)]_{v(u)}) \Rightarrow I'$. (Take U to be $\{u_1, u_2, u_3\} \cup MemLocs$, let $N(u_1)$ be $E.Next(p)$, etc.)

1.5.2. $IPImp(p) \land \Box Inv(p) \Rightarrow \overline{PInit(p)} \land \Box[\overline{UNext(p)}]_{\overline{pv}}$

PROOF: 1.1–1.4 show that $IPImp(p)$ implies $\overline{PInit(p)}$ and that

$$\begin{aligned}
&Inv(p) \land [E.Next(p)]_e \land [C.Next(p)]_c \land [R.Next(p)]_r \land [M.RNext(p)]_m \\
&\quad \land (\forall\, l \in MemLocs : [\exists\, q \in PrIds : M.Write(q, l)]_{mem[l]}) \\
&\quad \land [HNext(p)]_{\langle c, r, m, h \rangle}
\end{aligned}$$

implies $[\overline{UNext(p)}]_{\overline{pv}}$. The result is now obtained from the laws of TLA, which allow us in general to infer $\Box I \land (\forall u \in U : \Box[N(u)]_{v(u)}) \Rightarrow \Box[M]_w$ from $I \land I' \land (\forall u \in U : [N(u)]_{v(u)}) \Rightarrow [M]_w$.

1.5.3. $IPImp(p) \land \Box Inv(p) \Rightarrow \overline{\mathrm{WF}_{pv}(RNext(p))} \land \overline{\mathrm{WF}_{pv}(Return(p))}$

PROOF: Proved in section 3.3 below.

1.5.4. Q.E.D.

PROOF: Step 1.5 (which asserts step 1) follows from 1.5.1–1.5.3 by propositional logic.

3.3 The Proof of Step 1.5.3

To complete the proof of step 1, we must prove 1.5.3, which shows that the specification's fairness properties are satisfied. We give an intuitive sketch of the proof. To prove $\overline{\mathrm{WF}_{pv}(RNext(p))}$, we must show that if $\overline{RNext(p)}$ is continuously enabled, then a $\overline{RNext(p)}$ step must eventually occur. To prove $\overline{\mathrm{WF}_{pv}(Return(p))}$, we must show that if $\overline{Return(p)}$ is continuously enabled, then a $\overline{Return(p)}$ step must eventually occur. The two actions are disabled in state $S1$. Therefore, to prove the two fairness properties, it suffices to show that, if any of $S2$–$S6$ ever holds, then $S1$ must eventually hold. It is clear from the diagram in figure 3 that this follows from the two conditions: (i) none of the predicates $S2$–$S5$ can hold forever and (ii) if $S6$ holds repeatedly, then $S1$ must eventually hold. The following implementation fairness properties imply condition (i):

- $\mathrm{WF}_c(C.Forward(p))$ implies that $S2$ cannot hold forever.

- $\mathrm{WF}_r(R.Next(p))$ implies that $S3$ cannot hold forever.

- $\mathrm{WF}_m(M.RNext(p))$ implies that if $S4 \wedge (result[p] = NotAResult)$ holds, then $S4 \wedge (result[p] \neq NotAResult)$ eventually holds, and $\mathrm{WF}_m(M.Return(p))$ then implies that $S5$ eventually holds.

- $\mathrm{WF}_r(R.Next(p))$ implies that $S5$ cannot hold forever.

Condition (ii) follows from $\mathrm{SF}_c(C.Reply(p))$, which implies that if $S6$ holds repeatedly, then $S1$ eventually holds. The proof rules of TLA have been designed expressly to formalize this style of informal reasoning. The formal proof is as follows.

1.5.3. $IPImp(p) \wedge \Box Inv(p) \Rightarrow \overline{\mathrm{WF}_{pv}(RNext(p))} \wedge \overline{\mathrm{WF}_{pv}(Return(p))}$

1.5.3.1. $IPImp(p) \wedge \Box Inv(p) \Rightarrow \Box\Diamond S1$

PROOF: Proved in section 3.4 below.

1.5.3.2. $S1 \Rightarrow \neg \mathrm{ENABLED}\, \overline{\langle RNext(p) \rangle_{\overline{pv}}}$

$\langle 5 \rangle 1$. $\langle \overline{RNext(p)} \rangle_{\overline{pv}} \Rightarrow \overline{E.Calling(p)}$

PROOF: From the definitions of $RNext(p)$, $Read(p)$, $Write(p)$, $Return(p)$, and $E.Return(p, v)$, which all imply $E.Calling(p)$.

$\langle 5 \rangle 2$. Q.E.D.

PROOF: The definition of $S1$ shows $S1 \Rightarrow \neg E.Calling(p)$. The assertion follows by step $\langle 5 \rangle 1$, because $\overline{E.Calling(p)}$ is $E.Calling(p)$.

1.5.3.3. $IPImp(p) \wedge \Box Inv(p) \Rightarrow \Box\Diamond\neg \mathrm{ENABLED}\, \langle \overline{RNext(p)} \rangle_{\overline{pv}}$

PROOF: Step 1.5.3.2 implies

$\Box\Diamond S1 \Rightarrow \Box\Diamond\neg \mathrm{ENABLED}\, \langle \overline{RNext(p)} \rangle_{\overline{pv}}$

by the laws of temporal logic, which allow to infer $\Box\Diamond P \Rightarrow \Box\Diamond Q$ from $P \Rightarrow Q$, for any formulas P and Q. The assertion follows by step 1.5.3.1.

1.5.3.4. $S1 \Rightarrow \neg \mathrm{ENABLED}\, \overline{\langle Return(p) \rangle_{\overline{pv}}}$

PROOF: By the definition of $\overline{RNext(p)}$, we have $\langle \overline{Return(p)} \rangle_{\overline{pv}} \Rightarrow \langle \overline{RNext(p)} \rangle_{\overline{pv}}$. The assertion follows by step 1.5.3.2 and the monotonicity of ENABLED .

1.5.3.5. $IPImp(p) \wedge \Box Inv(p) \Rightarrow \Box\Diamond\neg \mathrm{ENABLED}\, \langle \overline{Return(p)} \rangle_{\overline{m}}$

PROOF: By the same argument as in the proof of step 1.5.3.3, using step 1.5.3.4 instead of step 1.5.3.2.

1.5.3.6. Q.E.D.

PROOF: From steps 1.5.3.3 and 1.5.3.5, since

$\mathrm{WF}_v(A) \equiv \Box\Diamond\neg \mathrm{ENABLED}\, \langle A \rangle_v \vee \Box\Diamond\langle A \rangle_v$

3.4 The proof of step 1.5.3.1

It remains to prove step 1.5.3.1, which asserts that predicate $S1$ holds infinitely often. As explained in section 3.3, we will prove that (i) any of the predicates $S2$–$S5$ will eventually be followed by $S6$ and that (ii) if $S6$ holds repeatedly, then $S1$ must eventually hold. The

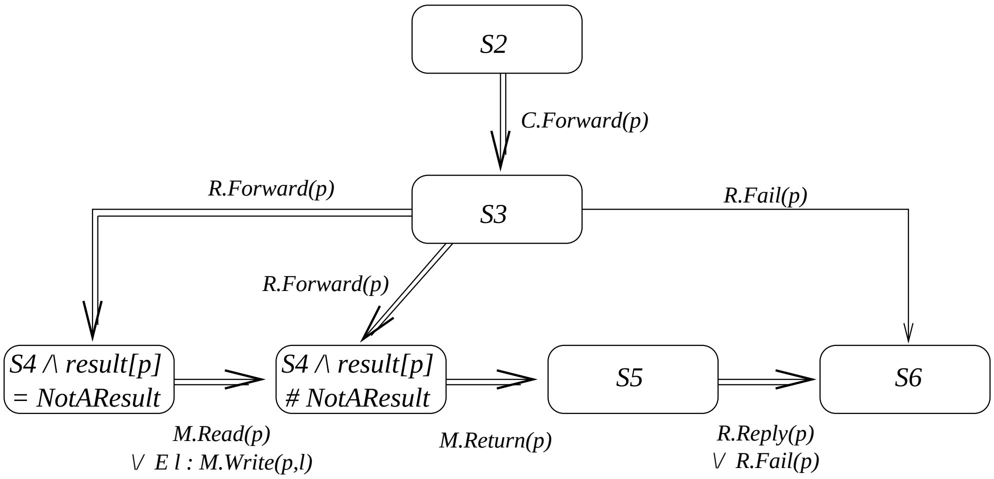


Figure 5: The leads-to lattice for the liveness proof.

proof of condition (i) is based on the diagram of figure 5 (which is a verification diagram in the style of [3]) and the weak fairness conditions of the implementation. Condition (ii) follows from $SF_c(C.Reply(p))$. Formally, the proof is as follows.

LET: $IPNext(p) \triangleq \land [E.Next(p)]_e$
$\land [C.Next(p)]_c \land [R.Next(p)]_r \land [M.RNext(p)]_m$
$\land \forall\, l \in MemLocs : [\exists\, q \in PrIds : M.Write(q,l)]_{mem[l]}$
$\land [HNext(p)]_{\langle c,r,m,h \rangle}$

1.5.3.1. $IPImp(p) \land \Box Inv(p) \Rightarrow \Box\Diamond S1$

$\langle 5 \rangle 1.$ $IPImp(p) \Rightarrow (S2 \lor S3 \lor S4 \lor S5 \lor S6 \leadsto S6)$

$\langle 6 \rangle 1.$ $IPImp(p) \Rightarrow (S2 \leadsto S3)$

$\langle 7 \rangle 1.$ $S2 \land IPNext(p) \Rightarrow S2' \lor S3'$

$\langle 8 \rangle 1.$ CASE: UNCHANGED $\langle e, c, r, m, h \rangle$

PROOF: The case assumption and lemma 8 imply $S2 \Rightarrow S2'$.

$\langle 8 \rangle 2.$ CASE: $\neg$UNCHANGED $\langle e, c, r, m, h \rangle$

PROOF: From assertion 2 of step 1.2, which implies

$S2 \land IPNext(p) \land \neg \text{UNCHANGED } \langle e, c, r, m, h \rangle \Rightarrow S3'$

$\langle 8 \rangle 3.$ Q.E.D.

PROOF: Steps $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$.

$\langle 7 \rangle 2.$ $S2 \land IPNext(p) \land \langle C.Forward(p) \rangle_c \Rightarrow S3'$

PROOF: From assertion 2 of step 1.2, observing that $\langle C.Forward(p) \rangle_c$ implies $\neg$UNCHANGED $\langle e, c, r, m, h \rangle$.

$\langle 7 \rangle 3.$ $S2 \Rightarrow$ ENABLED $\langle C.Forward(p) \rangle_c$

$\langle 8\rangle 1.$ $S2 \Rightarrow \wedge\ C.Snd.Calling(p) \wedge (cstate[p] = \text{“A”})$
$\wedge\ \neg C.Rcv.Calling(p)$

PROOF: Immediate from the definitions of $S2$ and S.

$\langle 8\rangle 2.$ $C.Snd.Calling(p) \wedge (cstate[p] = \text{“A”}) \wedge \neg C.Rcv.Calling(p)$
$\Rightarrow$ ENABLED $C.Forward(p)$

PROOF: From the definitions of $C.Forward(p)$ and $C.Rcv.Call(p, v)$, since the sets of state variables contained in $C.Rcv.caller(p)$, $C.Snd.rtrner(p)$, and $cstate[p]$ are pairwise disjoint.

$\langle 8\rangle 3.$ $C.Forward(p) \Rightarrow \langle C.Forward(p)\rangle_c$

PROOF: From the definition of $C.Forward(p)$, which implies $cstate'[p] \neq cstate[p]$ and therefore $c' \neq c$ by the definitions of c and $C.vars(p)$.

$\langle 8\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 8\rangle 1$ and $\langle 8\rangle 2$ imply $S2 \Rightarrow$ ENABLED $C.Forward(p)$. The assertion follows by step $\langle 8\rangle 3$ and the monotonicity of ENABLED .

$\langle 7\rangle 4.$ Q.E.D.

PROOF: By the TLA rule WF1, steps $\langle 7\rangle 1$–$\langle 7\rangle 3$ imply
$\Box IPNext(p) \wedge \mathrm{WF}_c(C.Forward(p)) \Rightarrow (S2 \leadsto S3)$
The assertion follows by the definition of $IPImp(p)$ and the fact that $\Box$ distributes over conjunction.

$\langle 6\rangle 2.$ $IPImp(p) \Rightarrow (S3 \leadsto S4 \vee S6)$

$\langle 7\rangle 1.$ $S3 \wedge IPNext(p) \Rightarrow S3' \vee S4' \vee S6'$

$\langle 8\rangle 1.$ CASE: UNCHANGED $\langle e, c, r, m, h\rangle$

PROOF: The case assumption and lemma 8 imply $S3 \Rightarrow S3'$.

$\langle 8\rangle 2.$ CASE: $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$

PROOF: From assertion 3 of step 1.2, which implies
$S3 \wedge IPNext(p) \wedge \neg\text{UNCHANGED } \langle e, c, r, m, h\rangle \Rightarrow S4' \vee S6'$

$\langle 8\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 8\rangle 1$ and $\langle 8\rangle 2$.

$\langle 7\rangle 2.$ $S3 \wedge IPNext(p) \wedge \langle R.Next(p)\rangle_r \Rightarrow S4' \vee S6'$

PROOF: From assertion 3 of step 1.2, observing that $\langle R.Next(p)\rangle_r$ implies $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$.

$\langle 7\rangle 3.$ $S3 \Rightarrow$ ENABLED $\langle R.Next(p)\rangle_r$

$\langle 8\rangle 1.$ $S3 \Rightarrow \neg R.Rcv.Calling(p) \wedge R.Snd.Calling(p)$

PROOF: Immediate from the definition of $S3$.

$\langle 8\rangle 2.$ $\neg R.Rcv.Calling(p) \wedge R.Snd.Calling(p) \Rightarrow$ ENABLED $R.Fail(p)$

PROOF: From the definitions of $R.Fail(p)$ and $R.Snd.Return(p, v)$, since the sets of state variables contained in $R.Snd.rtrner(p)$, $R.Rcv.caller(p)$, and $rstate[p]$ are pairwise disjoint.

$\langle 8\rangle 3.$ $R.Fail(p) \Rightarrow \langle R.Next(p)\rangle_r$

PROOF: The definition of $R.Next(p)$ shows that $R.Fail(p)$ implies $R.Next(p)$. Moreover, lemma 2 proves $R.Snd.rtrner(p)' \neq R.Snd.rtrner(p)$, which in turn implies $r' \neq r$ by the definitions of r and $R.vars(p)$.

$\langle 8 \rangle 4$. Q.E.D.

PROOF: Steps $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$ imply $S3 \Rightarrow$ ENABLED $R.Fail(p)$. The assertion follows by step $\langle 8 \rangle 3$ and the monotonicity of ENABLED .

$\langle 7 \rangle 4$. Q.E.D.

PROOF: By the TLA rule WF1, steps $\langle 7 \rangle 1$–$\langle 7 \rangle 3$ imply

$\Box IPNext(p) \wedge \mathrm{WF}_r(R.Next(p)) \Rightarrow (S3 \leadsto S4 \vee S6)$

The assertion follows by the definition of $IPImp(p)$ and the fact that $\Box$ distributes over conjunction.

$\langle 6 \rangle 3$. $IPImp(p)$

$\Rightarrow ((S4 \wedge result[p] = NotAResult) \leadsto (S4 \wedge result[p] \neq NotAResult))$

$\langle 7 \rangle 1$. $result[p] = NotAResult \Rightarrow \neg M.Return(p)$

PROOF: Immediate from the definition of $M.Return(p)$.

$\langle 7 \rangle 2$. $S4 \wedge M.Read(p, l) \Rightarrow result'[p] \neq NotAResult$

PROOF: The definition of $M.Read(p, l)$ implies

$M.Read(p, l) \Rightarrow \vee\ (l \in MemLocs) \wedge result'[p] = mem[l]$
$\qquad\qquad\qquad\quad \vee\ result'[p] = \text{“}\mathsf{BadArg}\text{”}$

By the definitions of $S4$ and S, we conclude

$S4 \wedge M.Read(p, l) \Rightarrow result'[p] \in MemVals \cup \{\text{“}\mathsf{BadArg}\text{”}\}$

The assertion now follows from the definition of $NotAResult$ and the axioms of set theory, which imply the existence of some $v \notin U$, for any U.

$\langle 7 \rangle 3$. $M.Write(p, l, v) \Rightarrow result'[p] \neq NotAResult$

PROOF: The definition of $M.Write(p, l, v)$ implies

$M.Write(p, l, v) \Rightarrow result'[p] \in \{\text{“}\mathsf{OK}\text{”}, \text{“}\mathsf{BadArg}\text{”}\}$

The assertion follows from the definition of $NotAResult$ and the axioms of set theory, which imply the existence of some $v \notin U$, for any U.

$\langle 7 \rangle 4$. $S4 \wedge (result[p] = NotAResult) \wedge IPNext(p) \Rightarrow S4'$

$\langle 8 \rangle 1$. CASE: UNCHANGED $\langle e, c, r, m, h \rangle$

PROOF: The case assumption and lemma 8 imply $S4 \Rightarrow S4'$.

$\langle 8 \rangle 2$. CASE: $\neg$UNCHANGED $\langle e, c, r, m, h \rangle$

PROOF: From assertion 4 of step 1.2, which implies

$S4 \wedge IPNext(p) \wedge \neg\text{UNCHANGED } \langle e, c, r, m, h \rangle$
$\Rightarrow S4' \vee M.Return(p)$

and step $\langle 7 \rangle 1$.

$\langle 8 \rangle 3$. Q.E.D.

PROOF: Steps $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$.

$\langle 7 \rangle 5$. $S4 \wedge (result[p] = NotAResult) \wedge IPNext(p) \wedge \langle M.RNext(p) \rangle_m$
$\Rightarrow S4' \wedge result[p] \neq NotAResult$

PROOF: Step ⟨7⟩1 and assertion 4 of step 1.2 imply

$S4 \wedge (result[p] = NotAResult) \wedge IPNext(p) \wedge \langle M.RNext(p) \rangle_m$
$\Rightarrow M.Read(p) \vee \exists\, l : M.Write(p, l)$

(since $\langle M.RNext(p) \rangle_m$ implies $\neg$UNCHANGED $\langle e, c, r, m, h \rangle$). The assertion follows by steps ⟨7⟩2, ⟨7⟩3, and ⟨7⟩4.

⟨7⟩6. $S4 \wedge (result[p] = NotAResult) \Rightarrow$ ENABLED $\langle M.RNext(p) \rangle_m$

⟨8⟩1. $S4 \Rightarrow M.E.Calling(p) \wedge (rmCh[p].arg \in LegalArgs)$

PROOF: The definitions of $S4$ and S imply

$$S4 \Rightarrow \wedge\ M.E.Calling(p) \\ \wedge\ rmCh[p].arg = R.RelayArg(p) \\ \wedge\ memCh[p].arg \in LegalArgs$$

The assertion follows by lemma 7, which implies

$S4 \Rightarrow R.RelayArg(p) = memCh[p].arg$

⟨8⟩2. $M.E.Calling(p) \wedge rmCh[p].arg = \langle \text{“Read”}, l \rangle$
$\Rightarrow$ ENABLED $M.Read(p, l)$

PROOF: From the definition of $M.Read(p, l)$, since the state variable *result* does not occur in $E.rtrner(p)$.

⟨8⟩3. $M.E.Calling(p) \wedge rmCh[p].arg = \langle \text{“Write”}, l, v \rangle$
$\Rightarrow$ ENABLED $M.Write(p, l, v)$

PROOF: From the definition of $M.Write(p, l, v)$, since the state variables *result* and *mem* do not occur in $E.rtrner(p)$.

⟨8⟩4. $S4 \wedge (result[p] = NotAResult)$
$\Rightarrow (M.Read(p, l) \vee M.Write(p, l, v) \Rightarrow \langle M.RNext(p) \rangle_m)$

PROOF: By the definition of $M.RNext(p)$, we obviously have

$M.Read(p, l) \vee M.Write(p, l, v) \Rightarrow M.RNext(p)$

Moreover, steps ⟨7⟩2 and ⟨7⟩3 imply

$S4 \wedge (result[p] = NotAResult)$
$\Rightarrow (M.Read(p, l) \vee M.Write(p, l, v) \Rightarrow result'[p] \neq result[p])$

The assertion follows by the definitions of m and $M.pvars(p)$.

⟨8⟩5. Q.E.D.

PROOF: Steps ⟨8⟩1–⟨8⟩3 and the definition of *LegalArgs* imply

$$S4 \Rightarrow \vee\ \exists\, l : \text{ENABLED } M.Read(p, l) \\ \vee\ \exists\, l, v : \text{ENABLED } M.Write(p, l, v)$$

Since existential quantification commutes with ENABLED (which is itself defined by existential quantification over primed variables), the assertion follows by first-order reasoning from step ⟨8⟩4 and the monotonicity of ENABLED .

⟨7⟩7. Q.E.D.

PROOF: By the TLA rule WF1, steps ⟨7⟩4–⟨7⟩6 imply

$\Box IPNext(p) \wedge \mathrm{WF}_r(M.RNext(p))$
$\Rightarrow ((S4 \wedge result[p] = NotAResult) \leadsto (S4 \wedge result[p] \neq NotAResult))$

The assertion follows by the definition of $IPImp(p)$ and the fact that $\Box$ distributes over conjunction.

$\langle 6 \rangle 4.$ $IPImp(p) \Rightarrow ((S4 \wedge result[p] \neq NotAResult) \leadsto S5)$

$\langle 7 \rangle 1.$ $S4 \wedge IPNext(p) \Rightarrow S4' \vee S5'$

$\langle 8 \rangle 1.$ CASE: $\text{UNCHANGED } \langle e, c, r, m, h \rangle$

PROOF: The case assumption and lemma 8 imply $S4 \Rightarrow S4'$.

$\langle 8 \rangle 2.$ CASE: $\neg \text{UNCHANGED } \langle e, c, r, m, h \rangle$

PROOF: From assertion 4 of step 1.2, which implies

$$S4 \wedge IPNext(p) \wedge \neg \text{UNCHANGED } \langle e, c, r, m, h \rangle \Rightarrow S4' \vee S5'$$

$\langle 8 \rangle 3.$ Q.E.D.

PROOF: Steps $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$.

$\langle 7 \rangle 2.$ $M.Return(p) \Rightarrow \neg(M.Read(p) \vee \exists\, l : M.Write(p, l))$

PROOF: The definition of $M.Return(p)$ and lemma 2 imply

$$M.Return(p) \Rightarrow \neg \text{UNCHANGED } M.E.rtrner(p)$$

On the other hand, the definitions of $M.Read(p)$ and $M.Write(p, l)$ both imply $\text{UNCHANGED } M.E.rtrner(p)$.

$\langle 7 \rangle 3.$ $S4 \wedge IPNext(p) \wedge \langle M.Return(p) \rangle_m \Rightarrow S5'$

PROOF: Assertion 4 of step 1.2 proves

$$\begin{aligned} & S4 \wedge IPNext(p) \wedge \langle M.Return(p) \rangle_m \\ & \Rightarrow S5' \vee M.Read(p) \vee \exists\, l : M.Write(p, l) \end{aligned}$$

since $\langle M.Return(p) \rangle_m$ implies $\neg \text{UNCHANGED } \langle e, c, r, m, h \rangle$. The assertion follows by step $\langle 7 \rangle 2$.

$\langle 7 \rangle 4.$ $S4 \wedge result[p] \neq NotAResult \Rightarrow \text{ENABLED } \langle M.Return(p) \rangle_m$

$\langle 8 \rangle 1.$ $S4 \Rightarrow M.E.Calling(p)$

PROOF: Immediate from the definitions of $S4$ and S.

$\langle 8 \rangle 2.$ $M.E.Calling(p) \wedge result[p] \neq NotAResult$
$\Rightarrow \text{ENABLED } M.Return(p)$

PROOF: From the definitions of $M.Return(p)$ and $M.E.Return(p, v)$.

$\langle 8 \rangle 3.$ $M.Return(p) \Rightarrow \langle M.Return(p) \rangle_m$

PROOF: The definition of $M.Return(p)$ implies $M.E.Return(p)$. Using lemma 2, we conclude

$$M.Return(p) \Rightarrow M.E.rtrner(p)' \neq M.E.rtrner(p)$$

The assertion follows by the definitions of m and $M.pvars(p)$.

$\langle 8 \rangle 4.$ Q.E.D.

PROOF: Steps $\langle 8 \rangle 1$ and $\langle 8 \rangle 2$ imply

$$S4 \wedge result[p] \neq NotAResult \Rightarrow \text{ENABLED } M.Return(p)$$

The assertion follows by step $\langle 8 \rangle 3$ and the monotonicity of ENABLED .

$\langle 7 \rangle 5.$ Q.E.D.

PROOF: By the TLA rule WF1, steps $\langle 7\rangle 1$, $\langle 7\rangle 3$, and $\langle 7\rangle 4$ imply

$\Box IPNext(p) \land \mathrm{WF}_m(M.Return(p))$
$\Rightarrow ((S4 \land result[p] \neq NotAResult) \leadsto S5)$

The assertion follows by the definition of $IPImp(p)$ and the fact that $\Box$ distributes over conjunction.

$\langle 6\rangle 5.$ $IPImp(p) \Rightarrow (S5 \leadsto S6)$

$\langle 7\rangle 1.$ $S5 \land IPNext(p) \Rightarrow S5' \lor S6'$

$\langle 8\rangle 1.$ CASE: UNCHANGED $\langle e, c, r, m, h\rangle$

PROOF: The case assumption and lemma 8 imply $S5 \Rightarrow S5'$.

$\langle 8\rangle 2.$ CASE: $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$

PROOF: From assertion 5 of step 1.2, which implies

$S5 \land IPNext(p) \land \neg \text{UNCHANGED } \langle e, c, r, m, h\rangle \Rightarrow S6'$

$\langle 8\rangle 3.$ Q.E.D.

PROOF: Steps $\langle 8\rangle 1$ and $\langle 8\rangle 2$.

$\langle 7\rangle 2.$ $S5 \land IPNext(p) \land \langle R.Next(p)\rangle_r \Rightarrow S6'$

PROOF: From assertion 5 of step 1.2, observing that $\langle R.Next(p)\rangle_r$ implies $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$.

$\langle 7\rangle 3.$ $S5 \Rightarrow \text{ENABLED } \langle R.Next(p)\rangle_r$

$\langle 8\rangle 1.$ $S5 \Rightarrow \neg R.Rcv.Calling(p) \land R.Snd.Calling(p) \land rstate[p] = \text{“B”}$

PROOF: Immediate from the definition of $S3$.

$\langle 8\rangle 2.$ $\neg R.Rcv.Calling(p) \land R.Snd.Calling(p) \land rstate[p] = \text{“B”}$
$\Rightarrow \text{ENABLED } R.Reply(p)$

PROOF: From the definitions of $R.Reply(p)$ and $R.Snd.Return(p, v)$, since the sets of state variables contained in $R.Snd.rtrner(p)$, $R.Rcv.caller(p)$, and $rstate[p]$ are pairwise disjoint.

$\langle 8\rangle 3.$ $R.Reply(p) \Rightarrow \langle R.Next(p)\rangle_r$

PROOF: By the definition of $R.Next(p)$, we have obviously

$R.Reply(p) \Rightarrow R.Next(p)$

Moreover, the definition of $R.Reply(p)$ shows

$R.Reply(p) \Rightarrow rstate'[p] \neq rstate[p]$

which proves the assertion, using the definitions of r and $R.vars(p)$.

$\langle 8\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 8\rangle 1$ and $\langle 8\rangle 2$ imply $S5 \Rightarrow \text{ENABLED } R.Reply(p)$. The assertion follows by step $\langle 8\rangle 3$ and the monotonicity of ENABLED .

$\langle 7\rangle 4.$ Q.E.D.

PROOF: By the TLA rule WF1, steps $\langle 7\rangle 1$–$\langle 7\rangle 3$ imply

$\Box IPNext(p) \land \mathrm{WF}_r(R.Next(p)) \Rightarrow (S5 \leadsto S6)$

The assertion follows by the definition of $IPImp(p)$ and the fact that $\Box$ distributes over conjunction.

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

PROOF: Steps $\langle 6\rangle 1$–$\langle 6\rangle 5$ imply that the diagram shown in figure 5 represents a valid leads-to lattice of the memory implementation. The assertion follows by simple lattice-style reasoning.

$\langle 5\rangle 2.$ $IPImp(p) \Rightarrow (\Box\Diamond S6 \Rightarrow \Box\Diamond S1)$

$\langle 6\rangle 1.$ $S6 \Rightarrow \text{ENABLED}\ \langle C.Reply(p)\rangle_c$

$\langle 7\rangle 1.$ $S6 \Rightarrow \neg C.Rcv.Calling(p) \wedge cstate[p] = \text{“B”} \wedge C.Snd.Calling(p)$

PROOF: Immediate from the definitions of $S6$ and S.

$\langle 7\rangle 2.$ $\neg C.Rcv.Calling(p) \wedge cstate[p] = \text{“B”} \wedge C.Snd.Calling(p)$
$\Rightarrow \text{ENABLED}\ C.Reply(p)$

PROOF: From the definitions of $C.Reply(p)$ and $C.Snd.Return(p, v)$, since the sets of state variables contained in $C.Snd.rtrner(p)$, $C.Rcv.caller(p)$, and $cstate[p]$ are pairwise disjoint.

$\langle 7\rangle 3.$ $C.Reply(p) \Rightarrow \langle C.Reply(p)\rangle_c$

PROOF: By the definition of $C.Reply(p)$, which implies $cstate'[p] \neq cstate[p]$, and of c and $C.vars(p)$.

$\langle 7\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 7\rangle 1$ and $\langle 7\rangle 2$ prove $S6 \Rightarrow \text{ENABLED}\ C.Reply(p)$. The assertion follows by step $\langle 7\rangle 3$ and the monotonicity of ENABLED .

$\langle 6\rangle 2.$ $S6 \wedge IPNext(p) \wedge \langle C.Reply(p)\rangle_c \Rightarrow S1'$

$\langle 7\rangle 1.$ $C.Reply(p) \Rightarrow \neg C.Retry(p)$

PROOF: The definition of $C.Reply(p)$ implies $cstate'[p] \neq cstate[p]$. On the other hand, the definition of $C.Retry(p)$ implies UNCHANGED $cstate[p]$.

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

PROOF: Assertion 6 of step 1.2 proves

$$S6 \wedge IPNext(p) \wedge \langle C.Reply(p)\rangle_c \Rightarrow S1' \vee C.Retry(p)$$

since $\langle C.Reply(p)\rangle_c$ implies $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$. The assertion follows by step $\langle 7\rangle 1$.

$\langle 6\rangle 3.$ Q.E.D.

PROOF: By simple temporal reasoning, steps $\langle 6\rangle 1$ and $\langle 6\rangle 2$ imply

$$\Box\Diamond S6 \Rightarrow \Box\Diamond \text{ENABLED}\ \langle C.Reply(p)\rangle_c$$

and

$$\Box IPNext(p) \wedge \Box\Diamond S6 \wedge \Box\Diamond\langle C.Reply(p)\rangle_c \Rightarrow \Box\Diamond S1$$

The assertion follows by the definition of $IPImp(p)$, which implies $SF_c(C.Reply(p))$ and hence

$$\Box\Diamond \text{ENABLED}\ \langle C.Reply(p)\rangle_c \Rightarrow \Box\Diamond\langle C.Reply(p)\rangle_c$$

and the fact that $\Box$ distributes over conjunction.

$\langle 5\rangle 3.$ $Inv(p) \Rightarrow (\neg S1 \equiv S2 \vee S3 \vee S4 \vee S5 \vee S6)$

PROOF: By propositional logic from the definition of $Inv(p)$.

$\langle 5\rangle 4.$ Q.E.D.

PROOF: Steps $\langle 5\rangle 1$ and $\langle 5\rangle 3$ imply
$IPImp(p) \wedge \Box Inv(p) \Rightarrow (\neg S1 \leadsto S6)$
The laws of temporal logic imply
$(P \leadsto Q) \Rightarrow (\Box\Diamond P \Rightarrow \Box\Diamond Q)$
for any predicates P and Q. We conclude
$IPImp(p) \wedge \Box Inv(p) \Rightarrow (\Box\Diamond\neg S1 \Rightarrow \Box\Diamond S6)$
and therefore, using step $\langle 5\rangle 2$,
$IPImp(p) \wedge \Box Inv(p) \Rightarrow (\Box\Diamond\neg S1 \Rightarrow \Box\Diamond S1)$
By propositional logic and the duality of $\Diamond$ and $\Box$, this is equivalent to
$IPImp(p) \wedge \Box Inv(p) \Rightarrow (\Diamond\Box S1 \vee \Box\Diamond S1)$
which completes the proof, observing that $\Diamond\Box S1$ implies $\Box\Diamond S1$.

4 The Proof of Step 2

Finally, we must prove step 2. We now confess that, to simplify the exposition, we have structured the proof incorrectly. The proof of 2 requires steps 1.1–1.4 and step 1.5.1, for an arbitrary p in *PrIds*. Those steps should therefore be brought out either as a separate lemma, or as level-1 steps. Here, we violate the rules of structured proofs and use those steps directly in the proof of 2.

2.1. $(\forall\, q \in PrIds : IPImp(q) \wedge \Box Inv(q)) \Rightarrow \overline{MInit(l)}$
PROOF: By the assumption that $l \in MemLocs$, since $M.MInit(l)$ trivially implies $\overline{MInit(l)}$ (the two formulas are the same).

2.2. $\wedge\ [E.Next(p)]_e$
$\wedge\ [C.Next(p)]_c \wedge [R.Next(p)]_r \wedge [M.RNext(p)]_m$
$\wedge\ \forall\, l1 \in MemLocs : [\exists\, q1 \in PrIds : M.Write(q1, l1)]_{mem[l1]}$
$\wedge\ [HNext(p)]_{\langle c,r,m,h\rangle}$
$\wedge\ Inv(p) \wedge Inv(p)'$
$\wedge\ [M.Write(q, l)]_{mem[l]}$
$\Rightarrow [\overline{Write(q, l)}]_{\overline{mem[l]}}$
PROOF: $Inv(p) \wedge M.Write(p, l)$ implies $S4$, for any p. We consider two cases. (i) If $\neg$UNCHANGED $\langle e, c, r, m, h\rangle$ holds, then the result follows from part 4 of 1.2 and part 4b of 1.4. (ii) If UNCHANGED $\langle e, c, r, m, h\rangle$ holds, then $S4$ and $Inv(p)'$ imply $S4'$, and the result follows from part 4b of 1.4.

2.3. Q.E.D.
PROOF: 2.1, 2.2, and the laws of TLA show that
$(\forall\, q \in PrIds : IPImp(q) \wedge \Box Inv(q)) \Rightarrow \overline{MSpec(l)}$
The result then follows from 1.5.1.

5 Auxiliary lemmas

This section contains several simple lemmas, which are used in the lower-level proofs of memory implementation.

Lemma 1 $Call(p, v) \Rightarrow caller(p)' \neq caller(p)$

PROOF: The definitions of $Call(p, v)$ and $Calling(p)$ in module *ProcedureInterface* imply

$$ch[p].cbit' \neq ch[p].rbit = ch[p].cbit$$

The assertion follows by the definition of $caller(p)$, since tuples are equal iff all their components are equal.

Lemma 2 $Return(p, v) \Rightarrow rtrner(p)' \neq rtrner(p)$

PROOF: The definitions of $Return(p, v)$ and $Calling(p)$ in module *ProcedureInterface* imply

$$ch[p].rbit' = ch[p].cbit \neq ch[p].rbit$$

The assertion follows by the definition of $rtrner(p)$, since tuples are equal iff all their components are equal.

Lemma 3 $E.Next(p) \Rightarrow \neg E.Calling(p)$

PROOF: By predicate logic from the definitions of $E.Next(p)$ and $E.Call(p, a)$.

Lemma 4 $C.Next(p) \Rightarrow \begin{array}{l} \wedge\ \neg R.Snd.Calling(p) \\ \wedge\ E.Calling(p) \vee (cstate[p] = \text{"B"}) \end{array}$

1. $C.Forward(p) \Rightarrow \neg R.Snd.Calling(p)$
   PROOF: The definition of $C.Forward(p)$ implies
   $C.Forward(p) \Rightarrow R.Snd.Call(p, RelayArg(p))$
   The assertion follows by the definition of $Call(p, v)$ in module *ProcedureInterface*.
2. $C.Forward(p) \Rightarrow E.Calling(p)$
   PROOF: Definition of $C.Forward(p)$, since $C.Snd.Calling(p) \equiv E.Calling(p)$.
3. $C.Retry(p) \Rightarrow \neg R.Snd.Calling(p)$
   PROOF: From the definitions of $C.Retry(p)$ and $Call(p, v)$, since $C.Rcv.Calling(p) \equiv R.Snd.Calling(p)$.
4. $C.Retry(p) \Rightarrow (cstate[p] = \text{"B"})$
   PROOF: Definition of $C.Retry(p)$.
5. $C.Reply(p) \Rightarrow \neg R.Snd.Calling(p)$
   PROOF: Definition of $C.Reply(p)$, since $C.Rcv.Calling(p) \equiv R.Snd.Calling(p)$.
6. $C.Reply(p) \Rightarrow (cstate[p] = \text{"B"})$
   PROOF: Definition of $C.Reply(p)$.

7. Q.E.D.
   PROOF: From steps 1–6 and the definition of $C.Next(p)$ by propositional logic.

Lemma 5 $R.Next(p) \Rightarrow \wedge\ R.Snd.Calling(p)$
$\wedge\ \neg M.E.Calling(p) \vee (rstate[p] = \text{“A”})$

1. $R.Forward(p) \Rightarrow R.Snd.Calling(p)$
   PROOF: Definition of $R.Forward(p)$.
2. $R.Forward(p) \Rightarrow (rstate[p] = \text{“A”})$
   PROOF: Definition of $R.Forward(p)$.
3. $R.Reject(p) \Rightarrow R.Snd.Calling(p)$
   PROOF: Definitions of $R.Reject(p)$ and $Return(p, v)$.
4. $R.Reject(p) \Rightarrow (rstate[p] = \text{“A”})$
   PROOF: Definition of $R.Reject(p)$.
5. $R.Fail(p) \Rightarrow R.Snd.Calling(p)$
   PROOF: Definitions of $R.Fail(p)$ and $Return(p, v)$.
6. $R.Fail(p) \Rightarrow \neg M.E.Calling(p)$
   PROOF: Definition of $R.Fail(p)$, since $R.Rcv.Calling(p) \equiv M.E.Calling(p)$.
7. $R.Reply(p) \Rightarrow R.Snd.Calling(p)$
   PROOF: Definitions of $R.Reply(p)$ and $Return(p, v)$.
8. $R.Reply(p) \Rightarrow \neg M.E.Calling(p)$
   PROOF: Definition of $R.Reply(p)$, since $R.Rcv.Calling(p) \equiv M.E.Calling(p)$.
9. Q.E.D.
   PROOF: From steps 1–8 and the definition of $R.Next(p)$ by propositional logic.

Lemma 6 $M.RNext(p) \Rightarrow M.E.Calling(p)$

1. $M.Read(p) \Rightarrow M.E.Calling(p)$
   PROOF: Definition of $M.Read(p)$.
2. $(\exists\, l : M.Write(p, l)) \Rightarrow M.E.Calling(p)$
   PROOF: Definition of $M.Write(p, l)$.
3. $M.Return(p) \Rightarrow M.E.Calling(p)$
   PROOF: By the definition of $M.Return(p)$, we have
   $M.Return(p) \Rightarrow M.E.Return(p, result[p])$
   The assertion follows using the definition of $Return(p, v)$ in module *ProcedureInterface*.
4. Q.E.D.
   PROOF: Steps 1–3 and the definition of $M.RNext(p)$.

Lemma 7
ASSUME: 1. rc, cs, rs, rh : CONSTANT
2. $S(\mathrm{T}, \mathrm{T}, rc, cs, rs, rh)$

PROVE: $R.RelayArg(p) = memCh[p].arg$

1. $memCh[p].arg \in Seq(Vals \cup \{\text{"Read"}, \text{"Write"}\})$
   PROOF: Assumption 2 and the definition of S imply $memCh[p].arg \in LegalArgs$. The assertion follows by the definition of $LegalArgs$ in module $MemoryParameters$ and by the definition of $Seq(S)$ in module $Sequences$.
2. $crCh[p].arg = \langle \text{"RemoteCall"}, Head(memCh[p].arg), Tail(memCh[p].arg) \rangle$
   PROOF: Assumption 2 and the definition of S imply $crCh[p].arg = C.RelayArg(p)$. The assertion follows by the definition of $C.RelayArg(p)$, since $C.sndCh = memCh$.
3. ASSUME: 1. S, x : CONSTANT
   2. $x \in Seq(S)$
   3. $Len(x) > 1$

   PROVE: $x = \langle Head(x) \rangle \circ Tail(x)$

   3.1. $Len(\langle Head(x) \rangle) = 1$
   PROOF: Immediate by the definition of $Len(s)$ and the representation of tuples in TLA$^+$.

   3.2. $Len(x) - 1 \in Nat$
   PROOF: From assumption ·3.

   3.3. $Len(Tail(x)) = Len(x) - 1$
   PROOF: Step 3.2 and the definitions of $Tail(x)$ and $Len(Tail(x))$.

   3.4. DOMAIN $(\langle Head(x) \rangle \circ Tail(x)) = OneTo(Len(x))$
   PROOF: The definition of concatenation implies that
   DOMAIN $(\langle Head(x) \rangle \circ Tail(x)) = OneTo(Len(\langle Head(x) \rangle) + Len(Tail(x)))$
   The assertion follows from steps 3.1–3.3 by simple arithmetic.

   3.5. DOMAIN $x = OneTo(Len(x))$
   PROOF: Assumption ·2 implies the existence of some $n \in Nat$ such that DOMAIN $x = OneTo(n)$. The assertion follows by the definition of $Len(x)$.

   3.6. ASSUME: $i \in OneTo(Len(x))$
   PROVE: $(\langle Head(x) \rangle \circ Tail(x))[i] = x[i]$

   $\langle 3 \rangle$1. CASE: $i \leq 1$
   PROOF: Assumption 3 and the case assumption imply $i = 1$, which by step 3.1 entails $i = Len(\langle Head(x) \rangle)$. Hence, the definitions of concatenation and of $Head(x)$ yield
   $(\langle Head(x) \rangle \circ Tail(x))[1] = \langle Head(x) \rangle[1] = Head(x) = x[1]$

   $\langle 3 \rangle$2. CASE: $\neg(i \leq 1)$

   $\langle 4 \rangle$1. $i \in Nat$
   PROOF: Assumption 2, which implies $Len(x) \in Nat$, assumption 3, and the definition of $OneTo(n)$.

   $\langle 4 \rangle$2. $1 < i \leq Len(x)$

PROOF: Step $\langle 4\rangle 1$ and assumption 3 imply $1 \leq i \leq Len(x)$. The case assumption $\langle 3\rangle$ implies $i \neq 1$ and therefore the assertion.

$\langle 4\rangle 3$. $i - 1 \in OneTo(Len(x) - 1)$

PROOF: Steps $\langle 4\rangle 1$, $\langle 4\rangle 2$, the definition of $OneTo(n)$, and simple facts about natural numbers.

$\langle 4\rangle 4$. $(\langle Head(x)\rangle \circ Tail(x))[i] = Tail(x)[i-1]$

PROOF: The definition of concatenation and steps 3.4 and 3.5 imply $i \in$ DOMAIN $\langle Head(x)\rangle \circ Tail(x)$. Therefore, the assertion follows from the definition of $(\langle Head(x)\rangle \circ Tail(x))[i]$.

$\langle 4\rangle 5$. Q.E.D.

PROOF: Step $\langle 4\rangle 3$, the definition of $Tail(x)$, and simple arithmetic imply $Tail(x)[i-1] = x[(i-1)+1] = x[i]$. Therefore, the assertion follows by step $\langle 4\rangle 4$.

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

PROOF: Steps $\langle 3\rangle 1$ and $\langle 3\rangle 2$.

3.7. Q.E.D.

PROOF: Steps 3.4–3.6.

4. Q.E.D.

PROOF: From steps 1–3 by the definition of $R.RelayArg(p)$ and equational reasoning.

Lemma 8

ASSUME: 1. ec, cc, rc, cs, rs, rh : CONSTANT

2. $S(ec, cc, rc, cs, rs, rh)$

3. UNCHANGED $\langle e, c, r, m, h\rangle$

PROVE: $S(ec, cc, rc, cs, rs, rh)'$

PROOF: Assumption 2 and the definitions of S and of $E.Calling(p)$, $R.Snd.Calling(p)$, and $M.E.Calling(p)$, all of which contain only variables contained in $\langle e, c, r, m, h\rangle$, by assumption 3.

Lemma 9 $S1 \Rightarrow \neg S2 \wedge \neg S3 \wedge \neg S4 \wedge \neg S5 \wedge \neg S6$

PROOF: The definition of $S1$ implies $S1 \Rightarrow \neg E.Calling(p)$, whereas the definitions of Si for $i \in \{2, \ldots, 6\}$ imply $Si \Rightarrow E.Calling(p)$.

Lemma 10 $S2 \Rightarrow \neg S1 \wedge \neg S3 \wedge \neg S4 \wedge \neg S5 \wedge \neg S6$

PROOF: Similar to that of lemma 9.

Lemma 11 $S3 \Rightarrow \neg S1 \wedge \neg S2 \wedge \neg S4 \wedge \neg S5 \wedge \neg S6$

PROOF: Similar to that of lemma 9.

module *ProcedureInterface*

parameters
$PrIds, Args$: CONSTANT
ch : VARIABLE

$$caller(p) \triangleq \langle ch[p].arg, ch[p].cbit \rangle$$
$$rtrner(p) \triangleq \langle ch[p].res, ch[p].rbit \rangle$$
$$Calling(p) \triangleq ch[p].cbit \neq ch[p].rbit$$
$$\begin{aligned} Call(p, v) \triangleq\ & \land \neg Calling(p) \\ & \land ch[p].cbit' \neq ch[p].rbit \\ & \land ch[p].arg' = v \end{aligned}$$
$$\begin{aligned} Return(p, v) \triangleq\ & \land Calling(p) \\ & \land ch[p].rbit' = ch[p].cbit \\ & \land ch[p].res' = v \end{aligned}$$

$$LegalCaller \triangleq \forall\, p \in PrIds : \neg Calling(p) \land \Box[\exists\, a \in Args : Call(p, a)]_{caller(p)}$$
$$LegalReturner \triangleq \forall\, p \in PrIds : \Box[\exists\, v : Return(p, v)]_{rtrner(p)}$$

Figure 6: Module *ProcedureInterface*.

Lemma 12 $S4 \Rightarrow \neg S1 \land \neg S2 \land \neg S3 \land \neg S5 \land \neg S6$

PROOF: Similar to that of lemma 9.

Lemma 13 $S5 \Rightarrow \neg S1 \land \neg S2 \land \neg S3 \land \neg S4 \land \neg S6$

PROOF: Similar to that of lemma 9.

Lemma 14 $S6 \Rightarrow \neg S1 \land \neg S2 \land \neg S3 \land \neg S4 \land \neg S5$

PROOF: Similar to that of lemma 9.

6 References

[1] Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[2] Leslie Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600–608, August-September 1993.

module *MemoryParameters*

export *MemoryParameters, E*

parameters

$MemLocs, MemVals, InitVal, Vals, PrIds$: CONSTANT

$memCh$: VARIABLE

assumption

$$ParamAssump \triangleq \begin{array}{l} \land\ MemLocs \cup MemVals \subseteq Vals \\ \land\ \{\text{“BadArg”}, \text{“MemFailure”}\} \cap MemVals = \{\} \\ \land\ InitVal \in MemVals \end{array}$$

$$LegalArgs \triangleq (\{\text{“Read”}\} \times Vals) \cup (\{\text{“Write”}\} \times Vals \times Vals)$$

include *ProcedureInterface* **as** E **with** $ch \leftarrow memCh, Args \leftarrow LegalArgs$

Figure 7: Module *MemoryParameters*.

[3] Zohar Manna and Amir Pnueli. Temporal verification diagrams. In *Proc. Int. Symp. on Theoretical Aspects of Computer Software*, volume 789 of LNCS, pages 726–765. Springer-Verlag, 1994.

module *Memory*

import *MemoryParameters*

export *Memory, E*

module *Inner*

parameters

$mem, result$: VARIABLE

$$
\begin{array}{lll}
NotAResult & \triangleq & \textsc{choose}\ v : \\
 & & \quad v \notin \{\text{“OK”}, \text{“BadArg”}, \text{“MemFailure”}\} \cup MemVals \\
MInit(l) & \triangleq & mem[l] = InitVal \\
PInit(p) & \triangleq & result[p] = NotAResult \\
Read(p) & \triangleq & \exists\, l : \land\ E.Calling(p) \\
 & & \qquad \land\ memCh[p].arg = \langle \text{“Read”}, l \rangle \\
 & & \qquad \land\ result'[p] = \textbf{if } l \in MemLocs \textbf{ then } mem[l] \\
 & & \qquad \qquad \qquad \qquad \qquad \qquad \textbf{else } \text{“BadArg”} \\
 & & \qquad \land\ \textsc{unchanged}\ E.rtrner(p) \\
Write(p, l) & \triangleq & \exists\, v : \land\ E.Calling(p) \\
 & & \qquad \land\ memCh[p].arg = \langle \text{“Write”}, l, v \rangle \\
 & & \qquad \land\ \lor\ \land\ (l \in MemLocs) \land (v \in MemVals) \\
 & & \qquad \qquad \quad \land\ mem'[l] = v \\
 & & \qquad \qquad \quad \land\ result'[p] = \text{“OK”} \\
 & & \qquad \quad \lor\ \land\ \neg((l \in MemLocs) \land (v \in MemVals)) \\
 & & \qquad \qquad \quad \land\ result'[p] = \text{“BadArg”} \\
 & & \qquad \qquad \quad \land\ \textsc{unchanged}\ mem[l] \\
 & & \qquad \land\ \textsc{unchanged}\ E.rtrner(p) \\
Fail(p) & \triangleq & \land\ E.Calling(p) \\
 & & \land\ result'[p] = \text{“MemFailure”} \\
 & & \land\ \textsc{unchanged}\ E.rtrner(p) \\
Return(p) & \triangleq & \land\ result[p] \neq NotAResult \\
 & & \land\ result'[p] = NotAResult \\
 & & \land\ E.Return(p, result[p]) \\
RNext(p) & \triangleq & Read(p) \lor (\exists\, l : Write(p, l)) \lor Return(p) \\
UNext(p) & \triangleq & RNext(p) \lor Fail(p) \\
pvars(p) & \triangleq & \langle E.rtrner(p), result[p] \rangle \\
RPSpec(p) & \triangleq & \land\ PInit(p) \\
 & & \land\ \Box[RNext(p)]_{pvars(p)} \\
 & & \land\ \mathrm{WF}_{pvars(p)}(RNext(p)) \ \land\ \mathrm{WF}_{pvars(p)}(Return(p)) \\
UPSpec(p) & \triangleq & \land\ PInit(p) \\
 & & \land\ \Box[UNext(p)]_{pvars(p)} \\
 & & \land\ \mathrm{WF}_{pvars(p)}(RNext(p)) \ \land\ \mathrm{WF}_{pvars(p)}(Return(p)) \\
MSpec(l) & \triangleq & MInit(l) \land \Box[\exists\, p \in PrIds : Write(p, l)]_{mem[l]} \\
IRSpec & \triangleq & (\forall\, p \in PrIds : RPSpec(p)) \ \land\ (\forall\, l \in MemLocs : MSpec(l)) \\
IUSpec & \triangleq & (\forall\, p \in PrIds : UPSpec(p)) \ \land\ (\forall\, l \in MemLocs : MSpec(l))
\end{array}
$$

$$
\begin{array}{lll}
RSpec & \triangleq & \boldsymbol{\exists}\, mem, result : Inner.IRSpec \\
USpec & \triangleq & \boldsymbol{\exists}\, mem, result : Inner.IUSpec
\end{array}
$$

Figure 8: Module *Memory*.

module *RPCParameters*

export *RPCParameters, Snd, Rcv*
import *Naturals, Sequences*

parameters $sndCh, rcvCh$: VARIABLE
$\quad Procs, ArgNum, Vals, PrIds$: CONSTANT

assumption $ParamAssump \triangleq ArgNum \in [Procs \rightarrow Nat]$

$LegalSndArgs \triangleq \{\text{“RemoteCall”}\} \times \text{STRING} \times Seq(Vals)$

$$LegalRcvArgs \triangleq \{s \in Seq(Procs \cup Vals) : \begin{array}{l} \wedge\ Len(s) > 0 \\ \wedge\ Head(s) \in Procs \\ \wedge\ Tail(s) \in Seq(Vals) \\ \wedge\ Len(s) = 1 + ArgNum[Head(s)]\ \} \end{array}$$

include *ProcedureInterface* **as** *Snd* **with** $ch \leftarrow sndCh, Args \leftarrow LegalSndArgs$
include *ProcedureInterface* **as** *Rcv* **with** $ch \leftarrow rcvCh, Args \leftarrow LegalRcvArgs$

Figure 9: Module *RPCParameters*.

module *Sequences*

import *Naturals*

$$\begin{array}{lcl}
OneTo(n) & \triangleq & \{i \in Nat : (1 \leq i) \wedge (i \leq n)\} \\
Seq(S) & \triangleq & \text{UNION } \{[OneTo(n) \rightarrow S] : n \in Nat\} \\
Len(s) & \triangleq & \text{CHOOSE } n : (n \in Nat) \wedge ((\text{DOMAIN } s) = OneTo(n)) \\
Head(s) & \triangleq & s[1] \\
Tail(s) & \triangleq & [i \in OneTo(Len(s) - 1) \mapsto s[i+1]] \\
(s) \circ (t) & \triangleq & [i \in OneTo(Len(s) + Len(t)) \mapsto \textbf{if } i \leq Len(s) \\
 & & \qquad \textbf{then } s[i] \\
 & & \qquad \textbf{else } t[i - Len(s)]\ ]
\end{array}$$

Figure 10: Module *Sequences*.

module *RPC*

import *RPCParameters, Naturals, Sequences*
export *RPC, Snd, Rcv*

module *Inner*

parameters
rstate : VARIABLE

$$
\begin{array}{lcl}
Init(p) & \triangleq & (rstate[p] = \text{“A”}) \land \neg Rcv.Calling(p) \\
RelayArg(p) & \triangleq & \langle sndCh[p].arg[2] \rangle \circ sndCh[p].arg[3] \\
Forward(p) & \triangleq & \land\ Snd.Calling(p) \land (rstate[p] = \text{“A”}) \\
 & & \land\ RelayArg(p) \in LegalRcvArgs \\
 & & \land\ Rcv.Call(p, RelayArg(p)) \\
 & & \land\ rstate'[p] = \text{“B”} \\
 & & \land\ \text{UNCHANGED } Snd.rtrner(p) \\
Reject(p) & \triangleq & \land\ rstate[p] = \text{“A”} \\
 & & \land\ RelayArg(p) \notin LegalRcvArgs \\
 & & \land\ Snd.Return(p, \text{“BadCall”}) \\
 & & \land\ \text{UNCHANGED } \langle rstate[p], Rcv.caller(p) \rangle \\
Fail(p) & \triangleq & \land\ \neg Rcv.Calling(p) \\
 & & \land\ Snd.Return(p, \text{“RPCFailure”}) \\
 & & \land\ rstate'[p] = \text{“A”} \\
 & & \land\ \text{UNCHANGED } Rcv.caller(p) \\
Reply(p) & \triangleq & \land\ \neg Rcv.Calling(p) \land (rstate[p] = \text{“B”}) \\
 & & \land\ Snd.Return(p, rcvCh[p].res) \\
 & & \land\ rstate'[p] = \text{“A”} \\
 & & \land\ \text{UNCHANGED } Rcv.caller(p) \\
Next(p) & \triangleq & Forward(p) \lor Reject(p) \lor Fail(p) \lor Reply(p) \\
vars(p) & \triangleq & \langle rstate[p], Snd.rtrner(p), Rcv.caller(p) \rangle \\
ISpec & \triangleq & \forall p \in PrIds : Init(p) \land \Box[Next(p)]_{vars(p)} \land \mathrm{WF}_{vars(p)}(Next(p))
\end{array}
$$

$$Spec \triangleq \exists\, rstate : Inner.ISpec$$

Figure 11: The specification of the RPC component.

module *MemClerkParameters*

export *MemClerkParameters, Snd, Rcv*

parameters $sndCh, rcvCh$: VARIABLE
$PrIds, Vals$: CONSTANT

$$LegalSndArgs \triangleq (\{\text{"Read"}\} \times Vals) \cup (\{\text{"Write"}\} \times Vals \times Vals)$$

$$LegalRcvArgs \triangleq \{\text{"RemoteCall"}\} \times \{\text{"Read"}, \text{"Write"}\} \times Seq(Vals)$$

include *ProcedureInterface* **as** *Snd* **with** $ch \leftarrow sndCh, Args \leftarrow LegalSndArgs$
include *ProcedureInterface* **as** *Rcv* **with** $ch \leftarrow rcvCh, Args \leftarrow LegalRcvArgs$

Figure 12: Module *MemClerkParameters*.

module *MemClerk*

import *MemClerkParameters, Sequences*

module *Inner*

parameters

$cstate$: VARIABLE

$$Init(p) \triangleq (cstate[p] = \text{“A”}) \land \neg Rcv.Calling(p)$$

$$RelayArg(p) \triangleq \langle \text{“RemoteCall”}, Head(sndCh[p].arg), Tail(sndCh[p].arg \rangle$$

$$ReplyVal(p) \triangleq \textbf{if } rcvCh[p].res = \text{“RPCFailure”} \textbf{ then } \text{“MemFailure”} \textbf{ else } rcvCh[p].res$$

$$\begin{aligned} Forward(p) \triangleq\ & \land Snd.Calling(p) \land (cstate[p] = \text{“A”}) \\ & \land Rcv.Call(p, RelayArg(p)) \\ & \land cstate'[p] = \text{“B”} \\ & \land \text{UNCHANGED } Snd.rtrner(p) \end{aligned}$$

$$\begin{aligned} Retry(p) \triangleq\ & \land (cstate[p] = \text{“B”}) \land (rcvCh[p].res = \text{“RPCFailure”}) \\ & \land Rcv.Call(p, RelayArg(p)) \\ & \land \text{UNCHANGED } \langle cstate[p], Snd.rtrner(p) \rangle \end{aligned}$$

$$\begin{aligned} Reply(p) \triangleq\ & \land \neg Rcv.Calling(p) \land (cstate[p] = \text{“B”}) \\ & \land Snd.Return(p, ReplyVal(p)) \\ & \land cstate'[p] = \text{“A”} \\ & \land \text{UNCHANGED } Rcv.caller(p) \end{aligned}$$

$$Next(p) \triangleq Forward(p) \lor Retry(p) \lor Reply(p)$$

$$vars(p) \triangleq \langle cstate[p], Snd.rtrner(p), Rcv.caller(p) \rangle$$

$$\begin{aligned} ISpec \triangleq\ & \forall p \in PrIds : \\ & \quad \land Init(p) \land \Box[Next(p)]_{vars(p)} \\ & \quad \land \mathrm{WF}_{vars(p)}(Forward(p)) \land \mathrm{SF}_{vars(p)}(Reply(p)) \end{aligned}$$

$$Spec \triangleq \exists\, cstate : Inner.ISpec$$

Figure 13: The component specification of the memory clerk.