MODULE *BPConProof*

This module specifies a *Byzantine Paxos* algorithm–a version of *Paxos* in which failed acceptors and leaders can be malicious. It is an abstraction and generalization of the Castro-Liskov algorithm in

author = "Miguel *Castro* and *Barbara Liskov*", title = "Practical byzantine fault tolerance and proactive
recovery",
journal = *ACM* Transactions on Computer Systems,
volume = 20,
number = 4, year = 2002, pages = "398–461"

EXTENDS *Integers*, *FiniteSets*, *FiniteSetTheorems*, *TLAPS*

The sets *Value* and *Ballot* are the same as in the Voting and *PConProof* specs.

CONSTANT *Value*

$Ballot \triangleq Nat$

As in module *PConProof*, we define *None* to be an unspecified value that is not an element of *Value*.

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

We pretend that which acceptors are good and which are malicious is specified in advance. Of course, the algorithm executed by the good acceptors makes no use of which acceptors are which. Hence, we can think of the sets of good and malicious acceptors as "prophecy constants" that are used only for showing that the algorithm implements the *PCon* algorithm.

We can assume that a maximal set of acceptors are bad, since a bad acceptor is allowed to do anything–including acting like a good one.

The basic idea is that the good acceptors try to execute the *Paxos* consensus algorithm, while the bad acceptors may try to prevent them.

We do not distinguish between faulty and non-faulty leaders. Safety must be preserved even if all leaders are malicious, so we allow any leader to send any syntactically correct message at any time. (In an implementation, syntactically incorrect messages are simply ignored by non-faulty acceptors and have no effect.) Assumptions about leader behavior are required only for liveness.

CONSTANTS *Acceptor*, — The set of good (non-faulty) acceptors.
FakeAcceptor, — The set of possibly malicious (faulty) acceptors.
ByzQuorum,

A *Byzantine* quorum is set of acceptors that includes a quorum of good ones. In the case that there are $2f+1$ good acceptors and f bad ones, a *Byzantine* quorum is any set of $2f+1$ acceptors.

WeakQuorum

A weak quorum is a set of acceptors that includes at least one good one. If there are f bad acceptors, then a weak quorum is any set of f+1 acceptors.

We define *ByzAcceptor* to be the set of all real or fake acceptors.

$ByzAcceptor \triangleq Acceptor \cup FakeAcceptor$

As in the *Paxos* consensus algorithm, we assume that the set of ballot numbers and -1 is disjoint from the set of all (real and fake) acceptors.

$\text{ASSUME } BallotAssump \triangleq (Ballot \cup \{-1\}) \cap ByzAcceptor = \{\}$

The following are the assumptions about acceptors and quorums that are needed to ensure safety of our algorithm.

$$\begin{aligned}
&\text{ASSUME } BQA \triangleq \\
&\quad \land Acceptor \cap FakeAcceptor = \{\} \\
&\quad \land \forall Q \in ByzQuorum : Q \subseteq ByzAcceptor \\
&\quad \land \forall Q1, Q2 \in ByzQuorum : Q1 \cap Q2 \cap Acceptor \neq \{\} \\
&\quad \land \forall Q \in WeakQuorum : \land Q \subseteq ByzAcceptor \\
&\qquad\qquad\qquad\qquad\qquad\quad \land Q \cap Acceptor \neq \{\}
\end{aligned}$$

The following assumption is not needed for safety, but it will be needed to ensure liveness.

$$\begin{aligned}
&\text{ASSUME } BQLA \triangleq \\
&\quad \land \exists Q \in ByzQuorum : Q \subseteq Acceptor \\
&\quad \land \exists Q \in WeakQuorum : Q \subseteq Acceptor
\end{aligned}$$

We now define the set *BMessage* of all possible messages.

$1aMessage \triangleq [type : \{\text{"1a"}\}, \; bal : Ballot]$

Type $1a$ messages are the same as in module *PConProof*.

$1bMessage \triangleq$

A $1b$ message serves the same function as a $1b$ message in ordinary Paxos, where the *mbal* and *mval* components correspond to the *mbal* and *mval* components in the $1b$ messages of *PConProof*. The $m2av$ component is set containing all records with *val* and *bal* components equal to the corresponding of components of a $2av$ message that the acceptor has sent, except containing for each *val* only the record corresponding to the $2av$ message with the highest *bal* component.

$$\begin{aligned}
&[type \; : \{\text{"1b"}\}, bal : Ballot, \\
&\; mbal \; : Ballot \cup \{-1\}, mval : Value \cup \{None\}, \\
&\; m2av : \text{SUBSET } [val : Value, bal : Ballot], \\
&\; acc : ByzAcceptor]
\end{aligned}$$

$1cMessage \triangleq$

Type $1c$ messages are the same as in *PConProof*.

$[type : \{\text{"1c"}\}, bal : Ballot, val : Value]$

$2avMessage \triangleq$

When an acceptor receives a $1c$ message, it relays that message's contents to the other acceptors in a $2av$ message. It does this only for the first $1c$ message it receives for that ballot; it can receive a second $1c$ message only if the leader is malicious, in which case it ignores that second $1c$ message.

$[type : \{\text{"2av"}\}, bal : Ballot, val : Value, acc : ByzAcceptor]$

$2bMessage \triangleq [type : \{\text{"2b"}\}, acc : ByzAcceptor, bal : Ballot, val : Value]$

$2b$ messages are the same as in ordinary *Paxos*.

$BMessage \triangleq$
$1aMessage \cup 1bMessage \cup 1cMessage \cup 2avMessage \cup 2bMessage$

We will need the following simple fact about these sets of messages.

LEMMA $BMessageLemma \triangleq$

$$\forall m \in BMessage : \wedge (m \in 1aMessage) \equiv (m.type = \text{"1a"}) \wedge (m \in 1bMessage) \equiv (m.type = \text{"1b"}) \wedge (m \in 1cMessage) \equiv (m.type = \text{"1c"}) \wedge (m \in 2avMessage) \equiv (m.type = \text{"2av"}) \wedge (m \in 2bMessage) \equiv (m.type = \text{"2b"})$$

$\langle 1 \rangle 1.$ $\wedge \forall m \in 1aMessage : m.type = \text{"1a"}$
$\wedge \forall m \in 1bMessage : m.type = \text{"1b"}$
$\wedge \forall m \in 1cMessage : m.type = \text{"1c"}$
$\wedge \forall m \in 2avMessage : m.type = \text{"2av"}$
$\wedge \forall m \in 2bMessage : m.type = \text{"2b"}$
BY DEF $1aMessage$, $1bMessage$, $1cMessage$, $2avMessage$, $2bMessage$

$\langle 1 \rangle 2.$ QED
BY $\langle 1 \rangle 1$ DEF $BMessage$

****

We now give the algorithm. The basic idea is that the set *Acceptor* of real acceptors emulate an execution of the *PCon* algorithm with Acceptor as its set of acceptors. Of course, they must do that without knowing which of the other processes in *ByzAcceptor* are real acceptors and which are fake acceptors. In addition, they don't know whether a leader is behaving according to the *PCon* algorithm or if it is malicious.

The main idea of the algorithm is that, before performing an action of the *PCon* algorithm, a good acceptor determines that this action is actually enabled in that algorithm. Since an action is enabled by the receipt of one or more messages, the acceptor has to determine that the enabling messages are legal *PCon* messages. Because algorithm *PCon* allows a $1a$ message to be sent at any time, the only acceptor action whose enabling messages must be checked is the $Phase2b$ action. It is enabled iff the appropriate $1c$ message and $2a$ message are legal. The $1c$ message is legal iff the leader has received the necessary $1b$ messages. The acceptor therefore maintains a set of $1b$ messages that it knows have been sent, and checks that those $1b$ messages enable the sending of the $1c$ message.

A $2a$ message is legal in the *PCon* algorithm iff (i) the corresponding $1c$ message is legal and (ii) it is the only $2a$ message that the leader sends. In the *BPCon* algorithm, there are no explicit $2a$ messages. They are implicitly sent by the acceptors when they send enough $2av$ messages.

We leave unspecified how an acceptor discovers what $1b$ messages have been sent. In the Castro-Liskov algorithm, this is done by having acceptors relay messages sent by other acceptors. An acceptor knows that a $1b$ message has been sent if it receives it directly or else receives a copy from a weak *Byzantine* quorum of acceptors. A (non-malicious) leader must determine what $1b$ messages acceptors know about so it chooses a value so that a quorum of acceptors will act on its $Phase1c$ message and cause that value to be chosen. However, this is necessary only for liveness, so we ignore this for now.

In other implementations of our algorithm, the leader sends along with the $1c$ message a proof that the necessary $1b$ messages have been sent. The easiest way to do this is to have acceptors digitally sign their $1b$ messages, so a copy of the message proves that it has been sent (by the acceptor indicated in the message's *acc* field). The necessary proofs can also be constructed using only message authenticators (like the ones used in the Castro-Liskov algorithm); how this is done is described elsewhere.

In the abstract algorithm presented here, which we call *BPCon*, we do not specify how acceptors learn what $1b$ messages have been sent. We simply introduce a variable *knowsSent* such that *knowsSent*[*a*] represents the set of $1b$ messages that (good) acceptor a knows have been sent, and have an action that nondeterministically adds sent $1b$ messages to this set.

--algorithm *BPCon***{**

****

The variables:

maxBal[*a*] = Highest ballot in which acceptor a has participated.

maxVBal[*a*] = Highest ballot in which acceptor a has cast a vote (sent a $2b$ message); or -1 if it hasn't cast a vote.

maxVVal[*a*] = *Value* acceptor a has voted for in ballot *maxVBal*[*a*], or *None* if *maxVBal*[*a*] = -1.

2*avSent*[*a*] = A set of records in [*val* : *Value*, *bal* : *Ballot*] describing the $2av$ messages that a has sent. A record is added to this set, and any element with the same *val* field (and lower *bal* field) removed when a sends a $2av$ message.

knownSent[*a*] = The set of $1b$ messages that acceptor a knows have been sent.

bmsgs = The set of all messages that have been sent. See the discussion of the *msgs* variable in module *PConProof* to understand our modeling of message passing.

****

variables $maxBal = [a \in Acceptor \mapsto -1],$
$maxVBal = [a \in Acceptor \mapsto -1],$
$maxVVal = [a \in Acceptor \mapsto None],$
$2avSent = [a \in Acceptor \mapsto \{\}],$
$knowsSent = [a \in Acceptor \mapsto \{\}],$
$bmsgs = \{\}$

define {

$sentMsgs(type, bal) \triangleq \{m \in bmsgs : m.type = type \land m.bal = bal\}$

$KnowsSafeAt(ac, b, v) \triangleq$

True for an acceptor *ac*, ballot *b*, and value *v* iff the set of $1b$ messages in *knowsSent*[*ac*] implies that value *v* is safe at ballot *b* in the *PaxosConsensus* algorithm being emulated by the good acceptors. To understand the definition, see the definition of *ShowsSafeAt* in module *PConProof* and recall (a) the meaning of the *mCBal* and *mCVal* fields of a $1b$ message and (*b*) that the set of real acceptors in a *ByzQuorum* forms a quorum of the *PaxosConsensus* algorithm.

$\text{LET } S \triangleq \{m \in knowsSent[ac] : m.bal = b\}$
$\text{IN } \lor \exists BQ \in ByzQuorum :$
$\quad \forall a \in BQ : \exists m \in S : \land m.acc = a$
$\quad\quad \land m.mbal = -1$

$$
\begin{array}{l}
\lor \exists c \in 0 \mathrel{..} (b-1) : \\
\quad \land \exists BQ \in ByzQuorum : \\
\qquad \forall a \in BQ : \exists m \in S : \land\ m.acc = a \\
\qquad\qquad\qquad\qquad\qquad\quad \land\ m.mbal \leq c \\
\qquad\qquad\qquad\qquad\qquad\quad \land\ (m.mbal = c) \Rightarrow (m.mval = v) \\
\quad \land \exists WQ \in WeakQuorum : \\
\qquad \forall a \in WQ : \\
\qquad\quad \exists m \in S : \land\ m.acc = a \\
\qquad\qquad\qquad\quad \land \exists r \in m.m2av : \land\ r.bal \geq c \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land\ r.val = v
\end{array}
$$

}

We now describe the processes' actions as macros.

The following two macros send a message and a set of messages, respectively. These macros are so simple that they're hardly worth introducing, but they do make the processes a little easier to read.

macro $SendMessage(m)\{bmsgs := bmsgs \cup \{m\}\}$
macro $SendSetOfMessages(S)\{bmsgs := bmsgs \cup S\}$

As in the *Paxos* consensus algorithm, a ballot *self* leader (good or malicious) can execute a $Phase1a$ ation at any time.

macro $Phase1a()\{SendMessage([type \mapsto \text{“1a”},\ bal \mapsto self])\}$

The acceptor's $Phase1b$ ation is similar to that of the *PaxosConsensus* algorithm.

macro $Phase1b(b)\{$
 when $(b > maxBal[self]) \land (sentMsgs(\text{“1a”},\ b) \neq \{\})$ **;**
 $maxBal[self] := b$ **;**
 $SendMessage([type \mapsto \text{“1b”},\ bal \mapsto b,\ acc \mapsto self,\ m2av \mapsto 2avSent[self],$
 $\qquad mbal \mapsto maxVBal[self],\ mval \mapsto maxVVal[self]])$
 }

A good ballot *self* leader can send a phase $1c$ message for value v if it knows that the messages in $knowsSent[a]$ for a *Quorum* of (good) acceptors imply that they know that v is safe at ballot *self* , and that they can convince any other acceptor that the appropriate $1b$ messages have been sent to that it will also know that v is safe at ballot *self* .

A malicious ballot *self* leader can send any phase $1c$ messages it wants (including one that a good leader could send). We prove safety with a $Phase1c$ ation that allows a leader to be malicious. To prove liveness, we will have to assume a good leader that sends only correct $1c$ messages.

As in the *PaxosConsensus* algorithm, we allow a $Phase1c$ action to send a set of $Phase1c$ messages. (This is not done in the Castro-Liskov algorithm, but seems natural in light of the *PaxosConsensus* algorithm.)

macro $Phase1c()\{$
  with $(S \in \text{SUBSET } [type : \{\text{“1c”}\},\ bal : \{self\},\ val : Value])\{$
  $\quad SendSetOfMessages(S)\}$
 }

If acceptor *self* receives a ballot b phase $1c$ message with value v, it relays v in a phase $2av$ message if

- it has not already sent a $2av$ message in this or a later ballot and
- the messages in $knowsSent[self]$ show it that v is safe at b in the non-Byzantine *Paxos* consensus algorithm being emulated.

macro $Phase2av(b)\{$
when $\wedge\ maxBal[self] \leq b$
$\wedge\ \forall\, r \in 2avSent[self] : r.bal < b$ **;**
We could just as well have used $r.bal \neq b$ in this condition.
with $(m \in \{ms \in sentMsgs(\text{"1c"}, b) : KnowsSafeAt(self, b, ms.val)\})\{$
$SendMessage([type \mapsto \text{"2av"}, bal \mapsto b, val \mapsto m.val, acc \mapsto self])$ **;**
$2avSent[self] := \{r \in 2avSent[self] : r.val \neq m.val\}$
$\cup \{[val \mapsto m.val, bal \mapsto b]\}$
$\}$ **;**
$maxBal[self] := b$ **;**
$\}$

Acceptor *self* can send a phase $2b$ message with value v if it has received phase $2av$ messages from a *Byzantine* quorum, which implies that a quorum of good acceptors assert that this is the first $1c$ message sent by the leader and that the leader was allowed to send that message. It sets $maxBal[self]$, $maxVBal[self]$, and $maxVVal[self]$ as in the non-Byzantine algorithm.

macro $Phase2b(b)\{$
when $maxBal[self] \leq b$ **;**
with $(v \in \{vv \in Value :$
$\exists\, Q \in ByzQuorum :$
$\forall\, aa \in Q :$
$\exists\, m \in sentMsgs(\text{"2av"}, b) : \wedge\ m.val = vv$
$\wedge\ m.acc = aa\})\{$
$SendMessage([type \mapsto \text{"2b"}, acc \mapsto self, bal \mapsto b, val \mapsto v])$ **;**
$maxVVal[self] := v$ **;**
$\}$ **;**
$maxBal[self] := b$ **;**
$maxVBal[self] := b$
$\}$

At any time, an acceptor can learn that some set of $1b$ messages were sent (but only if they atually were sent).

macro $LearnsSent(b)\{$
with $(S \in \text{SUBSET}\ sentMsgs(\text{"1b"}, b))\{$
$knowsSent[self] := knowsSent[self] \cup S$
$\}$
$\}$

A malicious acceptor *self* can send any acceptor message indicating that it is from itself. Since a malicious acceptor could allow other malicious processes to forge its messages, this action could represent the sending of the message by any malicious process.

```
macro FakingAcceptor(){
  with (m ∈ {mm ∈ 1bMessage ∪ 2avMessage ∪ 2bMessage :
              mm.acc = self}){
        SendMessage(m)
   }
 }
```

We combine these individual actions into a complete algorithm in the usual way, with separate process declarations for the acceptor, leader, and fake acceptor processes.

```
process (acceptor ∈ Acceptor){
  acc: while (TRUE){
         with (b ∈ Ballot){either Phase1b(b)or Phase2av(b)
                                   or Phase2b(b)or LearnsSent(b)}
  }
 }

process (leader ∈ Ballot){
  ldr: while (TRUE){
        either Phase1a()or Phase1c()
        }
 }

process (facceptor ∈ FakeAcceptor){
   facc : while (TRUE){FakingAcceptor()}
 }
}
```

Below is the TLA+ translation, as produced by the translator. (Some blank lines have been removed.)
****

BEGIN TRANSLATION

VARIABLES *maxBal*, *maxVBal*, *maxVVal*, *2avSent*, *knowsSent*, *bmsgs*

define statement

$sentMsgs(type, bal) \triangleq \{m \in bmsgs : m.type = type \land m.bal = bal\}$

```
KnowsSafeAt(ac, b, v) ≜
  LET S ≜ {m ∈ knowsSent[ac] : m.bal = b}
  IN   ∨ ∃ BQ ∈ ByzQuorum :
            ∀ a ∈ BQ : ∃ m ∈ S : ∧ m.acc = a
                                 ∧ m.mbal = −1
       ∨ ∃ c ∈ 0 .. (b − 1) :
            ∧ ∃ BQ ∈ ByzQuorum :
                 ∀ a ∈ BQ : ∃ m ∈ S : ∧ m.acc = a
                                      ∧ m.mbal ≤ c
                                      ∧ (m.mbal = c) ⇒ (m.mval = v)
            ∧ ∃ WQ ∈ WeakQuorum :
```

$$
\begin{array}{l}
\forall a \in WQ : \\
\quad \exists m \in S : \land\, m.acc = a \\
\qquad\qquad\quad \land\, \exists r \in m.m2av : \land\, r.bal \geq c \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad\;\; \land\, r.val = v
\end{array}
$$

$$vars \triangleq \langle maxBal,\ maxVBal,\ maxVVal,\ 2avSent,\ knowsSent,\ bmsgs \rangle$$

$$ProcSet \triangleq (Acceptor) \cup (Ballot) \cup (FakeAcceptor)$$

$Init \triangleq$ Global variables

$$
\begin{array}{l}
\land\, maxBal = [a \in Acceptor \mapsto -1] \\
\land\, maxVBal = [a \in Acceptor \mapsto -1] \\
\land\, maxVVal = [a \in Acceptor \mapsto None] \\
\land\, 2avSent \;\; = [a \in Acceptor \mapsto \{\}] \\
\land\, knowsSent = [a \in Acceptor \mapsto \{\}] \\
\land\, bmsgs = \{\}
\end{array}
$$

$$
\begin{array}{l}
acceptor(self) \triangleq \exists b \in Ballot : \\
\quad \lor\, \land\, (b > maxBal[self]) \land (sentMsgs(\text{“1a”},\ b) \neq \{\}) \\
\qquad \land\, maxBal' = [maxBal \text{ EXCEPT } ![self] = b] \\
\qquad \land\, bmsgs' = (bmsgs \cup \{([type \mapsto \text{“1b”},\ bal \mapsto b,\ acc \mapsto self,\ m2av \mapsto 2avSent[self], \\
\qquad\qquad\qquad\qquad\qquad mbal \mapsto maxVBal[self],\ mval \mapsto maxVVal[self]])\}) \\
\qquad \land\, \text{UNCHANGED } \langle maxVBal,\ maxVVal,\ 2avSent,\ knowsSent \rangle \\
\quad \lor\, \land\, \land\, maxBal[self] \leq b \\
\qquad\quad \land\, \forall r \in 2avSent[self] : r.bal < b \\
\qquad \land\, \exists m \in \{ms \in sentMsgs(\text{“1c”},\ b) : KnowsSafeAt(self,\ b,\ ms.val)\} : \\
\qquad\qquad \land\, bmsgs' = (bmsgs \cup \{([type \mapsto \text{“2av”},\ bal \mapsto b,\ val \mapsto m.val,\ acc \mapsto self])\}) \\
\qquad\qquad \land\, 2avSent' = [2avSent \text{ EXCEPT } ![self] = \{r \in 2avSent[self] : r.val \neq m.val\} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cup \{[val \mapsto m.val,\ bal \mapsto b]\}] \\
\qquad \land\, maxBal' = [maxBal \text{ EXCEPT } ![self] = b] \\
\qquad \land\, \text{UNCHANGED } \langle maxVBal,\ maxVVal,\ knowsSent \rangle \\
\quad \lor\, \land\, maxBal[self] \leq b \\
\qquad \land\, \exists v \in \{vv \in Value : \\
\qquad\qquad\qquad \exists Q \in ByzQuorum : \\
\qquad\qquad\qquad\quad \forall aa \in Q : \\
\qquad\qquad\qquad\qquad \exists m \in sentMsgs(\text{“2av”},\ b) : \land\, m.val = vv \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land\, m.acc = aa\} : \\
\qquad\qquad \land\, bmsgs' = (bmsgs \cup \{([type \mapsto \text{“2b”},\ acc \mapsto self,\ bal \mapsto b,\ val \mapsto v])\}) \\
\qquad\qquad \land\, maxVVal' = [maxVVal \text{ EXCEPT } ![self] = v] \\
\qquad \land\, maxBal' = [maxBal \text{ EXCEPT } ![self] = b] \\
\qquad \land\, maxVBal' = [maxVBal \text{ EXCEPT } ![self] = b] \\
\qquad \land\, \text{UNCHANGED } \langle 2avSent,\ knowsSent \rangle \\
\quad \lor\, \land\, \exists S \in \text{SUBSET } sentMsgs(\text{“1b”},\ b) : \\
\qquad\qquad knowsSent' = [knowsSent \text{ EXCEPT } ![self] = knowsSent[self] \cup S] \\
\qquad \land\, \text{UNCHANGED } \langle maxBal,\ maxVBal,\ maxVVal,\ 2avSent,\ bmsgs \rangle
\end{array}
$$

$leader(self) \triangleq \land \lor \land bmsgs' = (bmsgs \cup \{([type \mapsto \text{"1a"}, bal \mapsto self])\})$
$\quad\quad \lor \land \exists S \in \text{SUBSET } [type : \{\text{"1c"}\}, bal : \{self\}, val : Value] :$
$\quad\quad\quad bmsgs' = (bmsgs \cup S)$
$\quad \land \text{UNCHANGED } \langle maxBal, maxVBal, maxVVal, 2avSent, knowsSent \rangle$

$facceptor(self) \triangleq \land \exists m \in \{mm \in 1bMessage \cup 2avMessage \cup 2bMessage :$
$\quad\quad mm.acc = self\} :$
$\quad\quad bmsgs' = (bmsgs \cup \{m\})$
$\quad \land \text{UNCHANGED } \langle maxBal, maxVBal, maxVVal, 2avSent,$
$\quad\quad knowsSent \rangle$

$Next \triangleq (\exists self \in Acceptor : acceptor(self))$
$\quad \lor (\exists self \in Ballot : leader(self))$
$\quad \lor (\exists self \in FakeAcceptor : facceptor(self))$

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

END TRANSLATION

As in module *PConProof*, we now rewrite the next-state relation in a form more convenient for writing proofs.

$Phase1b(self, b) \triangleq$
$\quad \land (b > maxBal[self]) \land (sentMsgs(\text{"1a"}, b) \neq \{\})$
$\quad \land maxBal' = [maxBal \text{ EXCEPT } ![self] = b]$
$\quad \land bmsgs' = bmsgs \cup \{[type \mapsto \text{"1b"}, bal \mapsto b, acc \mapsto self,$
$\quad\quad m2av \mapsto 2avSent[self],$
$\quad\quad mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]\}$
$\quad \land \text{UNCHANGED } \langle maxVBal, maxVVal, 2avSent, knowsSent \rangle$

$Phase2av(self, b) \triangleq$
$\quad \land maxBal[self] \leq b$
$\quad \land \forall r \in 2avSent[self] : r.bal < b$
$\quad \land \exists m \in \{ms \in sentMsgs(\text{"1c"}, b) : KnowsSafeAt(self, b, ms.val)\} :$
$\quad\quad \land bmsgs' = bmsgs \cup$
$\quad\quad\quad \{[type \mapsto \text{"2av"}, bal \mapsto b, val \mapsto m.val, acc \mapsto self]\}$
$\quad\quad \land 2avSent' = [2avSent \text{ EXCEPT}$
$\quad\quad\quad ![self] = \{r \in 2avSent[self] : r.val \neq m.val\}$
$\quad\quad\quad\quad \cup \{[val \mapsto m.val, bal \mapsto b]\}]$
$\quad \land maxBal' = [maxBal \text{ EXCEPT } ![self] = b]$
$\quad \land \text{UNCHANGED } \langle maxVBal, maxVVal, knowsSent \rangle$

$Phase2b(self, b) \triangleq$
$\quad \land maxBal[self] \leq b$
$\quad \land \exists v \in \{vv \in Value :$
$\quad\quad \exists Q \in ByzQuorum :$

$\forall a \in Q :$

$\quad \exists m \in sentMsgs(\text{“2av”}, b) : \land\ m.val = vv$

$\qquad\qquad\qquad\qquad\qquad\qquad\quad \land\ m.acc = a\} :$

$\land\ bmsgs' = (bmsgs \cup$

$\qquad \{[type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]\})$

$\land\ maxVVal' = [maxVVal\ \text{EXCEPT}\ ![self] = v]$

$\land\ maxBal' = [maxBal\ \text{EXCEPT}\ ![self] = b]$

$\land\ maxVBal' = [maxVBal\ \text{EXCEPT}\ ![self] = b]$

$\land\ \text{UNCHANGED}\ \langle 2avSent, knowsSent \rangle$

$LearnsSent(self, b) \triangleq$

$\land\ \exists S \in \text{SUBSET}\ sentMsgs(\text{“1b”}, b) :$

$\qquad knowsSent' = [knowsSent\ \text{EXCEPT}\ ![self] = knowsSent[self] \cup S]$

$\land\ \text{UNCHANGED}\ \langle maxBal, maxVBal, maxVVal, 2avSent, bmsgs \rangle$

$Phase1a(self) \triangleq$

$\land\ bmsgs' = (bmsgs \cup \{[type \mapsto \text{“1a”}, bal \mapsto self]\})$

$\land\ \text{UNCHANGED}\ \langle maxBal, maxVBal, maxVVal, 2avSent, knowsSent \rangle$

$Phase1c(self) \triangleq$

$\land\ \exists S \in \text{SUBSET}\ [type : \{\text{“1c”}\}, bal : \{self\}, val : Value] :$

$\qquad bmsgs' = (bmsgs \cup S)$

$\land\ \text{UNCHANGED}\ \langle maxBal, maxVBal, maxVVal, 2avSent, knowsSent \rangle$

$FakingAcceptor(self) \triangleq$

$\land\ \exists m \in \{mm \in 1bMessage \cup 2avMessage \cup 2bMessage : mm.acc = self\} :$

$\qquad bmsgs' = (bmsgs \cup \{m\})$

$\land\ \text{UNCHANGED}\ \langle maxBal, maxVBal, maxVVal, 2avSent, knowsSent \rangle$

The following lemma describes how the next-state relation *Next* can be written in terms of the actions defined above.

LEMMA $NextDef \triangleq$

$Next \equiv \lor\ \exists self \in Acceptor :$

$\qquad\qquad \exists b \in Ballot : \lor\ Phase1b(self, b)$

$\qquad\qquad\qquad\qquad\qquad \lor\ Phase2av(self, b)$

$\qquad\qquad\qquad\qquad\qquad \lor\ Phase2b(self, b)$

$\qquad\qquad\qquad\qquad\qquad \lor\ LearnsSent(self, b)$

$\qquad\quad \lor\ \exists self \in Ballot : \lor\ Phase1a(self)$

$\qquad\qquad\qquad\qquad\qquad \lor\ Phase1c(self)$

$\qquad\quad \lor\ \exists self \in FakeAcceptor : FakingAcceptor(self)$

$\langle 1 \rangle 1.\ \forall self : acceptor(self) \equiv NextDef!2!1!(self)$

BY DEF $acceptor, Phase1b, Phase2av, Phase2b, LearnsSent$

$\langle 1 \rangle 2.\ \forall self : leader(self) \equiv NextDef!2!2!(self)$

BY DEF $leader, Phase1a, Phase1c$

$\langle 1 \rangle 3.\ \forall self : facceptor(self) \equiv NextDef!2!3!(self)$

BY DEF $facceptor, FakingAcceptor$

$\langle 1\rangle 4.$ QED
  BY $\langle 1\rangle 1$, $\langle 1\rangle 2$, $\langle 1\rangle 3$, *Zenon*
  DEF *Next*, *acceptor*, *leader*, *facceptor*

THE REFINEMENT MAPPING

We define a quorum to be the set of acceptors in a *Byzantine* quorum. The quorum assumption *QA* of module *PConProof*, which we here call *QuorumTheorem*, follows easily from the definition and assumption *BQA*.

$$Quorum \triangleq \{S \cap Acceptor : S \in ByzQuorum\}$$

THEOREM *QuorumTheorem* $\triangleq$

$$\land \forall Q1, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$$
$$\land \forall Q \in Quorum : Q \subseteq Acceptor$$

BY *BQA* DEF *Quorum*

We now define refinement mapping under which our algorithm implements the algorithm of module *PConProof*. First, we define the set *msgs* that implements the variable of the same name in *PConProof*. There are two non-obvious parts of the definition.

1. The $1c$ messages in *msgs* should just be the ones that are legal–that is, messages whose value is safe at the indicated ballot. The obvious way to define legality is in terms of $1b$ messages that have been sent. However, this has the effect that sending a $1b$ message can add both that $1b$ message and one or more $1c$ messages to *msgs*. Proving implementation under this refinement mapping would require adding a stuttering variable. Instead, we define the $1c$ message to be legal if the set of $1b$ messages that some acceptor knows were sent confirms its legality. Thus, those $1c$ messages are added to *msgs* by the *LearnsSent* ation, which has no other effect on the refinement mapping.

2. A $2a$ message is added to *msgs* when a quorum of acceptors have reacted to it by sending a $2av$ message.

$$msgsOfType(t) \triangleq \{m \in bmsgs : m.type = t\}$$

$$acceptorMsgsOfType(t) \triangleq \{m \in msgsOfType(t) : m.acc \in Acceptor\}$$

$$1bRestrict(m) \triangleq [type \mapsto \text{“1b”}, acc \mapsto m.acc, bal \mapsto m.bal, mbal \mapsto m.mbal, mval \mapsto m.mval]$$

$$1bmsgs \triangleq \{1bRestrict(m) : m \in acceptorMsgsOfType(\text{“1b”})\}$$

$$1cmsgs \triangleq \{m \in msgsOfType(\text{“1c”}) : \exists a \in Acceptor : KnowsSafeAt(a, m.bal, m.val)\}$$

$$\begin{aligned}2amsgs \triangleq \{m \in [type : \{\text{“2a”}\}, bal : Ballot, val : Value] : \\ \exists Q \in Quorum : \\ \forall a \in Q : \\ \exists m2av \in acceptorMsgsOfType(\text{“2av”}) : \\ \land m2av.acc = a \\ \land m2av.bal = m.bal \\ \land m2av.val = m.val\}\end{aligned}$$

$msgs \triangleq msgsOfType(\text{"1a"}) \cup 1bmsgs \cup 1cmsgs \cup 2amsgs$
$\quad \cup\ acceptorMsgsOfType(\text{"2b"})$

We now define $PmaxBal$, the state function with which we instantiate the variable $maxBal$ of $PConProof$. The reason we don't just instantiate it with the variable $maxBal$ is that $maxBal[a]$ can change when acceptor a performs a $Phase2av$ ation, which does not correspond to any acceptor action of the $PCon$ algorithm. We want $PmaxBal[a]$ to change only when a performs a $Phase1b$ or $Phase2b$ ation–that is, when it sends a $1b$ or $2b$ message. Thus, we define $PmaxBal[a]$ to be the largest bal field of all $1b$ and $2b$ messages sent by a .

To define $PmaxBal$, we need to define an operator $MaxBallot$ so that $MaxBallot(S)$ is the largest element of S if S is non-empty a finite set consisting of ballot numbers and possibly the value -1.

$MaxBallot(S) \triangleq$
$\quad \text{IF } S = \{\} \text{ THEN } -1$
$\qquad\qquad \text{ELSE CHOOSE } mb \in S : \forall x \in S : mb \geq x$

To prove that the CHOOSE in this definition actually does choose a maximum of S when S is nonempty, we need the following fact.

LEMMA $FiniteSetHasMax \triangleq$
$\quad \forall S \in \text{SUBSET } Int :$
$\qquad IsFiniteSet(S) \land (S \neq \{\}) \Rightarrow \exists max \in S : \forall x \in S : max \geq x$

$\langle 1 \rangle$.DEFINE $P(S) \triangleq S \subseteq Int \land S \neq \{\} \Rightarrow$
$\qquad\qquad \exists max \in S : \forall x \in S : max \geq x$

$\langle 1 \rangle 1.\ P(\{\})$
  OBVIOUS

$\langle 1 \rangle 2.$ ASSUME NEW T, NEW x, $P(T)$
  PROVE $P(T \cup \{x\})$
  BY $\langle 1 \rangle 2$

$\langle 1 \rangle 3.\ \forall S : IsFiniteSet(S) \Rightarrow P(S)$
  $\langle 2 \rangle$.HIDE DEF P
  $\langle 2 \rangle$.QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $FS\_Induction$, $IsaM(\text{"blast"})$

$\langle 1 \rangle$.QED BY $\langle 1 \rangle 3$, $Zenon$

Our proofs use this property of $MaxBallot$.

THEOREM $MaxBallotProp \triangleq$
  ASSUME NEW $S \in$ SUBSET $(Ballot \cup \{-1\})$,
  $IsFiniteSet(S)$
  PROVE IF $S = \{\}$ THEN $MaxBallot(S) = -1$
  ELSE $\land\ MaxBallot(S) \in S$
  $\land\ \forall x \in S : MaxBallot(S) \geq x$

$\langle 1 \rangle 1$.CASE $S = \{\}$
  BY $\langle 1 \rangle 1$ DEF $MaxBallot$

$\langle 1 \rangle 2$.CASE $S \neq \{\}$
  $\langle 2 \rangle$.PICK $mb \in S : \forall x \in S : mb \geq x$
    BY $\langle 1 \rangle 2$, $FiniteSetHasMax$ DEF $Ballot$
  $\langle 2 \rangle$.QED BY $\langle 1 \rangle 2$ DEF $MaxBallot$

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

We now prove a couple of lemmas about *MaxBallot*.

LEMMA $MaxBallotLemma1 \triangleq$
ASSUME NEW $S \in$ SUBSET $(Ballot \cup \{-1\})$,
$IsFiniteSet(S)$,
NEW $y \in S$, $\forall x \in S : y \geq x$
PROVE $y = MaxBallot(S)$
$\langle 1 \rangle 1.$ $\land\ MaxBallot(S) \in S$
$\land\ MaxBallot(S) \geq y$
BY $MaxBallotProp$
$\langle 1 \rangle 2$ $\land\ y \in Ballot \cup \{-1\}$
$\land\ y \geq MaxBallot(S)$
BY $MaxBallotProp$
$\langle 1 \rangle 3.$ $MaxBallot(S) \in Int \land y \in Int$
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, Isa DEF $Ballot$
$\langle 1 \rangle$.QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

LEMMA $MaxBallotLemma2 \triangleq$
ASSUME NEW $S \in$ SUBSET $(Ballot \cup \{-1\})$,
NEW $T \in$ SUBSET $(Ballot \cup \{-1\})$,
$IsFiniteSet(S)$, $IsFiniteSet(T)$
PROVE $MaxBallot(S \cup T) =$ IF $MaxBallot(S) \geq MaxBallot(T)$
THEN $MaxBallot(S)$ ELSE $MaxBallot(T)$
$\langle 1 \rangle 1.$ $\land\ MaxBallot(S) \in Ballot \cup \{-1\}$
$\land\ MaxBallot(T) \in Ballot \cup \{-1\}$
BY $MaxBallotProp$
$\langle 1 \rangle.$ $S \cup T \subseteq Int$
BY DEF $Ballot$
$\langle 1 \rangle 2.$CASE $MaxBallot(S) \geq MaxBallot(T)$
$\langle 2 \rangle$.SUFFICES ASSUME $T \neq \{\}$
PROVE $MaxBallot(S \cup T) = MaxBallot(S)$
BY $\langle 1 \rangle 2$, $Zenon$
$\langle 2 \rangle 1.$ $\land\ MaxBallot(T) \in T$
$\land\ \forall x \in T : MaxBallot(T) \geq x$
BY $MaxBallotProp$
$\langle 2 \rangle 2.$CASE $S = \{\}$
$\langle 3 \rangle 1.$ $MaxBallot(S) = -1$
BY $\langle 2 \rangle 2$ DEF $MaxBallot$
$\langle 3 \rangle 2.$ $MaxBallot(T) = -1$
BY $\langle 3 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 1$ DEF $Ballot$
$\langle 3 \rangle$.QED BY $\langle 2 \rangle 2$, $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$, $MaxBallotLemma1$, FS_Union
$\langle 2 \rangle 3.$CASE $S \neq \{\}$
$\langle 3 \rangle 1.$ $\land\ MaxBallot(S) \in S$
$\land\ \forall x \in S : MaxBallot(S) \geq x$
BY $\langle 2 \rangle 3$, $MaxBallotProp$
$\langle 3 \rangle 2.$ $\land\ MaxBallot(S) \in S \cup T$

$\land \forall x \in S \cup T : MaxBallot(S) \geq x$
BY ⟨3⟩1, ⟨2⟩1, ⟨1⟩2
⟨3⟩.QED BY ⟨3⟩2, *MaxBallotLemma*1, *FS_Union*, *Zenon*
⟨2⟩.QED BY ⟨2⟩2, ⟨2⟩3
⟨1⟩3.CASE $\neg(MaxBallot(S) \geq MaxBallot(T))$
⟨2⟩.SUFFICES ASSUME $S \neq \{\}$
PROVE $MaxBallot(S \cup T) = MaxBallot(T)$
BY ⟨1⟩3
⟨2⟩1. $\land MaxBallot(S) \in S$
$\land \forall x \in S : MaxBallot(S) \geq x$
BY *MaxBallotProp*
⟨2⟩2. $\land MaxBallot(S) < MaxBallot(T)$
$\land MaxBallot(T) \neq -1$
BY ⟨1⟩3, ⟨1⟩1 DEF *Ballot*
⟨2⟩3. $\land MaxBallot(T) \in T$
$\land \forall x \in T : MaxBallot(T) \geq x$
BY ⟨2⟩2, *MaxBallotProp*
⟨2⟩4. $\land MaxBallot(T) \in S \cup T$
$\land \forall x \in S \cup T : MaxBallot(T) \geq x$
BY ⟨2⟩3, ⟨2⟩2, ⟨2⟩1
⟨2⟩.QED BY ⟨2⟩4, *MaxBallotLemma*1, *FS_Union*, *Zenon*
⟨1⟩.QED BY ⟨1⟩2, ⟨1⟩3

We finally come to our definition of *PmaxBal*, the state function substituted for variable *maxBal* of module *PConProof* by our refinement mapping. We also prove a couple of lemmas about *PmaxBal*.

$1bOr2bMsgs \triangleq \{m \in bmsgs : m.type \in \{\text{“1b”}, \text{“2b”}\}\}$

$PmaxBal \triangleq [a \in Acceptor \mapsto MaxBallot(\{m.bal : m \in \{ma \in 1bOr2bMsgs : ma.acc = a\}\})]$

LEMMA *PmaxBalLemma*1 $\triangleq$
ASSUME NEW m,
$bmsgs' = bmsgs \cup \{m\}$,
$m.type \neq \text{“1b”} \land m.type \neq \text{“2b”}$
PROVE $PmaxBal' = PmaxBal$
BY *Zenon* DEF *PmaxBal*, $1bOr2bMsgs$

LEMMA *PmaxBalLemma*2 $\triangleq$
ASSUME NEW m,
$bmsgs' = bmsgs \cup \{m\}$,
NEW $a \in Acceptor$,
$m.acc \neq a$
PROVE $PmaxBal'[a] = PmaxBal[a]$

BY DEF $PmaxBal$, $1bOr2bMsgs$

Finally, we define the refinement mapping. As before, for any operator *op* defined in module *PConProof*, the following INSTANCE statement defines $P!op$ to be the operator obtained from *op* by the indicated substitutions, along with the implicit substitutions

$$\begin{aligned} &Acceptor \leftarrow Acceptor,\\ &Quorum \leftarrow Quorum\\ &Value \leftarrow Value\\ &maxVBal \leftarrow maxVBal\\ &maxVVal \leftarrow maxVVal\\ &msgs \quad \leftarrow msgs \end{aligned}$$

$P \triangleq$ INSTANCE $PConProof$ WITH $maxBal \leftarrow PmaxBal$

We now define the inductive invariant *Inv* used in our proof. It is defined to be the conjunction of a number of separate invariants that we define first, starting with the ever-present type-correctness invariant.

$$\begin{aligned} TypeOK \triangleq\ &\land maxBal \in [Acceptor \to Ballot \cup \{-1\}]\\ &\land 2avSent \in [Acceptor \to \text{SUBSET } [val : Value,\ bal : Ballot]]\\ &\land maxVBal \in [Acceptor \to Ballot \cup \{-1\}]\\ &\land maxVVal \in [Acceptor \to Value \cup \{None\}]\\ &\land knowsSent \in [Acceptor \to \text{SUBSET } 1bMessage]\\ &\land bmsgs \subseteq BMessage \end{aligned}$$

To use the definition of *PmaxBal*, we need to know that the set of $1b$ and $2b$ messages in *bmsgs* is finite. This is asserted by the following invariant. Note that the set *bmsgs* is not necessarily finite because we allow a $Phase1c$ action to send an infinite number of $1c$ messages.

$bmsgsFinite \triangleq IsFiniteSet(1bOr2bMsgs)$

The following lemma is used to prove the invariance of *bmsgsFinite*.

LEMMA $FiniteMsgsLemma \triangleq$
ASSUME NEW m, $bmsgsFinite$, $bmsgs' = bmsgs \cup \{m\}$
PROVE $bmsgsFinite'$
BY $FS\_AddElement$ DEF $bmsgsFinite$, $1bOr2bMsgs$

Invariant $1bInv1$ asserts that if (good) acceptor a has $mCBal[a] \neq -1$, then there is a $1c$ message for ballot $mCBal[a]$ and value $mCVal[a]$ in the emulated execution of algorithm *PCon*.

$$\begin{aligned} 1bInv1 \triangleq\ &\forall m \in bmsgs :\\ &\quad \land m.type = \text{"1b"}\\ &\quad \land m.acc \in Acceptor\\ &\quad \Rightarrow \forall r \in m.m2av :\\ &\qquad [type \mapsto \text{"1c"},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs \end{aligned}$$

Invariant $1bInv2$ asserts that an acceptor sends at most one $1b$ message for any ballot.

$$\begin{aligned} 1bInv2 \triangleq\ &\forall m1, m2 \in bmsgs :\\ &\quad \land m1.type = \text{"1b"}\\ &\quad \land m2.type = \text{"1b"}\\ &\quad \land m1.acc \in Acceptor \end{aligned}$$

$$\begin{array}{l} \land\ m1.acc = m2.acc \\ \land\ m1.bal\ = m2.bal \\ \Rightarrow m1 = m2 \end{array}$$

Invariant $2avInv1$ asserts that an acceptor sends at most one $2av$ message in any ballot.

$$2avInv1 \triangleq \forall\, m1,\, m2 \in bmsgs : \begin{array}{l} \land\ m1.type = \text{"2av"} \\ \land\ m2.type = \text{"2av"} \\ \land\ m1.acc \in Acceptor \\ \land\ m1.acc = m2.acc \\ \land\ m1.bal\ = m2.bal \\ \Rightarrow m1 = m2 \end{array}$$

Invariant $2avInv2$ follows easily from the meaning (and setting) of $2avSent$.

$$2avInv2 \triangleq \forall\, m \in bmsgs : \begin{array}{l} \land\ m.type = \text{"2av"} \\ \land\ m.acc \in Acceptor \\ \Rightarrow \exists\, r \in 2avSent[m.acc] : \begin{array}{l} \land\ r.val = m.val \\ \land\ r.bal \geq m.bal \end{array} \end{array}$$

Invariant $2avInv3$ asserts that an acceptor sends a $2av$ message only if the required $1c$ message exists in the emulated execution of algorithm $PConf$.

$$2avInv3 \triangleq \forall\, m \in bmsgs : \begin{array}{l} \land\ m.type = \text{"2av"} \\ \land\ m.acc \in Acceptor \\ \Rightarrow [type \mapsto \text{"1c"},\ bal \mapsto m.bal,\ val \mapsto m.val] \in msgs \end{array}$$

Invariant $maxBalInv$ is a simple consequence of the fact that an acceptor a sets $maxBal[a]$ to b whenever it sends a $1b$, $2av$, or $2b$ message in ballot b.

$$maxBalInv \triangleq \forall\, m \in bmsgs : \begin{array}{l} \land\ m.type \in \{\text{"1b"}, \text{"2av"}, \text{"2b"}\} \\ \land\ m.acc\ \in Acceptor \\ \Rightarrow m.bal \leq maxBal[m.acc] \end{array}$$

Invariant $accInv$ asserts some simple relations between the variables local to an acceptor, as well as the fact that acceptor a sets $maxCBal[a]$ to b and $maxCVal[a]$ to v only if there is a ballot-b $1c$ message for value c in the simulated execution of the $PCon$ algorithm.

$$accInv \triangleq \forall\, a \in Acceptor : \forall\, r \in 2avSent[a] : \begin{array}{l} \land\ r.bal \leq maxBal[a] \\ \land\ [type \mapsto \text{"1c"},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs \end{array}$$

Invariant $knowsSentInv$ simply asserts that for any acceptor a , $knowsSent[a]$ is a set of $1b$ messages that have actually been sent.

$$knowsSentInv \triangleq \forall\, a \in Acceptor : knowsSent[a] \subseteq msgsOfType(\text{"1b"})$$

$$Inv \triangleq$$

$TypeOK \land bmsgsFinite \land 1bInv1 \land 1bInv2 \land maxBalInv \land 2avInv1 \land 2avInv2$
$\land\, 2avInv3 \land accInv \land knowsSentInv$

We now prove some simple lemmas that are useful for reasoning about *PmaxBal*.

LEMMA $PMaxBalLemma3 \triangleq$
ASSUME $TypeOK,$
$bmsgsFinite,$
NEW $a \in Acceptor$
PROVE LET $S \triangleq \{m.bal : m \in \{ma \in bmsgs :$
$\land\, ma.type \in \{\text{“1b”}, \text{“2b”}\}$
$\land\, ma.acc = a\}\}$
IN $\land\, IsFiniteSet(S)$
$\land\, S \in$ SUBSET $Ballot$

$\langle 1 \rangle$ DEFINE $T \triangleq \{ma \in bmsgs : \land\, ma.type \in \{\text{“1b”}, \text{“2b”}\}$
$\land\, ma.acc = a\}$
$S \triangleq \{m.bal : m \in T\}$
$\langle 1 \rangle 1.\ IsFiniteSet(S)$
$\langle 2 \rangle 1.\ IsFiniteSet(T)$
BY FS_Subset DEF $bmsgsFinite, 1bOr2bMsgs$
$\langle 2 \rangle$.QED
BY $\langle 2 \rangle 1, FS\_Image, Isa$
$\langle 1 \rangle$.QED BY $\langle 1 \rangle 1, BMessageLemma$ DEF $1bMessage, 2bMessage, TypeOK$

LEMMA $PmaxBalLemma4 \triangleq$
ASSUME $TypeOK,$
$maxBalInv,$
$bmsgsFinite,$
NEW $a \in Acceptor$
PROVE $PmaxBal[a] \leq maxBal[a]$

$\langle 1 \rangle$ DEFINE $SM \triangleq \{ma \in bmsgs : \land\, ma.type \in \{\text{“1b”}, \text{“2b”}\}$
$\land\, ma.acc = a\}$
$S \triangleq \{ma.bal : ma \in SM\}$
$\langle 1 \rangle 1.\ PmaxBal[a] = MaxBallot(S)$
BY DEF $PmaxBal, 1bOr2bMsgs$
$\langle 1 \rangle 2.\ \land\, IsFiniteSet(S)$
$\land\, S \in$ SUBSET $Ballot$
BY $PMaxBalLemma3$
$\langle 1 \rangle 3.\ \forall b \in S : b \leq maxBal[a]$
BY DEF $maxBalInv$
$\langle 1 \rangle 4.$CASE $S = \{\}$
$\langle 2 \rangle 1.\ PmaxBal[a] = -1$
BY $\langle 1 \rangle 2, \langle 1 \rangle 1, \langle 1 \rangle 4, MaxBallotProp$
$\langle 2 \rangle$.QED
BY $\langle 2 \rangle 1$ DEF $Ballot, TypeOK$
$\langle 1 \rangle 5.$CASE $S \neq \{\}$

$\langle 2 \rangle 1.$ $MaxBallot(S) \in S$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 5$, *MaxBallotProp*, *Zenon*
$\langle 2 \rangle 2.$ QED
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $\langle 2 \rangle 1$
$\langle 1 \rangle 6.$ QED
BY $\langle 1 \rangle 4$, $\langle 1 \rangle 5$

LEMMA $PmaxBalLemma5 \triangleq$
ASSUME $TypeOK$, $bmsgsFinite$, NEW $a \in Acceptor$
PROVE $PmaxBal[a] \in Ballot \cup \{-1\}$
BY *PMaxBalLemma3*, *MaxBallotProp* DEF *PmaxBal*, *1bOr2bMsgs*

Now comes a bunch of useful lemmas.

We first prove that $P!NextDef$ is a valid theorem and give it the name *PNextDef*. This requires proving that the assumptions of module *PConProof* are satisfied by the refinement mapping. Note that $P!NextDef!:$ is an abbreviation for the statement of theorem $P!NextDef$ – that is, for the statement of theorem *NextDef* of module *PConProof* under the substitutions of the refinement mapping.

LEMMA $PNextDef \triangleq P!NextDef!:$
$\langle 1 \rangle 1.$ $P!QA$
BY *QuorumTheorem*
$\langle 1 \rangle 2.$ $P!BallotAssump$
BY *BallotAssump* DEF *Ballot*, $P!Ballot$, *ByzAcceptor*
$\langle 1 \rangle 3.$ QED
BY $P!NextDef$, $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, *NoSetContainsEverything*

For convenience, we define operators corresponding to subexpressions that appear in the definition of *KnowsSafeAt*.

$$KSet(a, b) \triangleq \{m \in knowsSent[a] : m.bal = b\}$$
$$KS1(S) \triangleq \exists BQ \in ByzQuorum : \forall a \in BQ : \exists m \in S : m.acc = a \land m.mbal = -1$$
$$\begin{aligned}KS2(v, b, S) \triangleq\ & \exists c \in 0 \,..\, (b-1) : \\ & \quad \land \exists BQ \in ByzQuorum : \forall a \in BQ : \\ & \qquad \exists m \in S : \land m.acc = a \\ & \qquad\qquad \land m.mbal \leq c \\ & \qquad\qquad \land (m.mbal = c) \Rightarrow (m.mval = v) \\ & \quad \land \exists WQ \in WeakQuorum : \forall a \in WQ : \\ & \qquad \exists m \in S : \land m.acc = a \\ & \qquad\qquad \land \exists r \in m.m2av : \land r.bal \geq c \\ & \qquad\qquad\qquad \land r.val = v\end{aligned}$$

The following lemma asserts the obvious relation between *KnowsSafeAt* and the top-level definitions *KS1*, *KS2*, and *KSet*. The second conjunct is, of course, the primed version of the first.

LEMMA $KnowsSafeAtDef \triangleq$

$\forall\, a,\, b,\, v :$
$\quad \land\, KnowsSafeAt(a,\, b,\, v) \equiv KS1(KSet(a,\, b)) \lor KS2(v,\, b,\, KSet(a,\, b))$
$\quad \land\, KnowsSafeAt(a,\, b,\, v)' \equiv KS1(KSet(a,\, b)') \lor KS2(v,\, b,\, KSet(a,\, b)')$
BY DEF $KnowsSafeAt,\, KSet,\, KS1,\, KS2$

LEMMA $MsgsTypeLemma \triangleq$
$\forall\, m \in msgs : \land\, (m.type = \text{"1a"}) \equiv (m \in msgsOfType(\text{"1a"}))$
$\qquad \land\, (m.type = \text{"1b"}) \equiv (m \in 1bmsgs)$
$\qquad \land\, (m.type = \text{"1c"}) \equiv (m \in 1cmsgs)$
$\qquad \land\, (m.type = \text{"2a"}) \equiv (m \in 2amsgs)$
$\qquad \land\, (m.type = \text{"2b"}) \equiv (m \in acceptorMsgsOfType(\text{"2b"}))$
BY DEF $msgsOfType,\, 1bmsgs,\, 1bRestrict,\, 1cmsgs,\, 2amsgs,\, acceptorMsgsOfType,\, msgs$

The following lemma is the primed version of *MsgsTypeLemma*. That is, its statement is just the statement of *MsgsTypeLemma* primed. It follows from *MsgsTypeLemma* by the meta-theorem that if we can prove a state-predicate F as a (top-level) theorem, then we can deduce F'. This is an instance of propositional temporal-logic reasoning. Alternatively the lemma could be proved using the same reasoning used for the unprimed version of the theorem.

LEMMA $MsgsTypeLemmaPrime \triangleq$
$\forall\, m \in msgs' : \land\, (m.type = \text{"1a"}) \equiv (m \in msgsOfType(\text{"1a"})')$
$\qquad \land\, (m.type = \text{"1b"}) \equiv (m \in 1bmsgs')$
$\qquad \land\, (m.type = \text{"1c"}) \equiv (m \in 1cmsgs')$
$\qquad \land\, (m.type = \text{"2a"}) \equiv (m \in 2amsgs')$
$\qquad \land\, (m.type = \text{"2b"}) \equiv (m \in acceptorMsgsOfType(\text{"2b"})')$
$\langle 1 \rangle 1.\ MsgsTypeLemma'$
  BY $MsgsTypeLemma,\, PTL$
$\langle 1 \rangle$.QED
  BY $\langle 1 \rangle 1$

The following lemma describes how *msgs* is changed by the actions of the algorithm.

LEMMA $MsgsLemma \triangleq$
$TypeOK \Rightarrow$
$\quad \land\ \ \forall\, self \in Acceptor,\, b \in Ballot :$
$\qquad Phase1b(self,\, b) \Rightarrow$
$\qquad\quad msgs' = msgs \cup$
$\qquad\qquad \{[type \mapsto \text{"1b"},\, acc \mapsto self,\, bal \mapsto b,$
$\qquad\qquad\quad mbal \mapsto maxVBal[self],\, mval \mapsto maxVVal[self]]\}$
$\quad \land\ \ \forall\, self \in Acceptor,\, b \in Ballot :$
$\qquad Phase2av(self,\, b) \Rightarrow$
$\qquad\quad \lor\, msgs' = msgs$
$\qquad\quad \lor\, \exists\, v \in Value :$
$\qquad\qquad \land\, [type \mapsto \text{"1c"},\, bal \mapsto b,\, val \mapsto v] \in msgs$
$\qquad\qquad \land\, msgs' = msgs \cup \{[type \mapsto \text{"2a"},\, bal \mapsto b,\, val \mapsto v]\}$
$\quad \land\ \ \forall\, self \in Acceptor,\, b \in Ballot :$
$\qquad Phase2b(self,\, b) \Rightarrow$
$\qquad\quad \exists\, v \in Value :$

$\land \exists Q \in ByzQuorum :$
$\quad \forall a \in Q :$
$\quad\quad \exists m \in sentMsgs(\text{"2av"}, b) : \land m.val = v$
$\quad\quad\quad \land m.acc = a$
$\land msgs' = msgs \cup$
$\quad \{[type \mapsto \text{"2b"}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$
$\land bmsgs' = bmsgs \cup$
$\quad \{[type \mapsto \text{"2b"}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$
$\land maxVVal' = [maxVVal \text{ EXCEPT } ![self] = v]$

$\land \quad \forall self \in Acceptor, b \in Ballot :$
$\quad LearnsSent(self, b) \Rightarrow$
$\quad\quad \exists S \in \text{SUBSET } \{m \in msgsOfType(\text{"1c"}) : m.bal = b\} :$
$\quad\quad\quad msgs' = msgs \cup S$

$\land \quad \forall self \in Ballot :$
$\quad Phase1a(self) \Rightarrow$
$\quad\quad msgs' = msgs \cup \{[type \mapsto \text{"1a"}, bal \mapsto self]\}$

$\land \quad \forall self \in Ballot :$
$\quad Phase1c(self) \Rightarrow$
$\quad\quad \exists S \in \text{SUBSET } [type : \{\text{"1c"}\}, bal : \{self\}, val : Value] :$
$\quad\quad\quad \land \forall m \in S :$
$\quad\quad\quad\quad \exists a \in Acceptor : KnowsSafeAt(a, m.bal, m.val)$
$\quad\quad\quad \land msgs' = msgs \cup S$

$\land \quad \forall self \in FakeAcceptor : FakingAcceptor(self) \Rightarrow msgs' = msgs$

$\langle 1 \rangle$ HAVE $TypeOK$

$\langle 1 \rangle 1.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$, $Phase1b(self, b)$
PROVE $msgs' = msgs \cup$
$\quad \{[type \mapsto \text{"1b"}, acc \mapsto self, bal \mapsto b,$
$\quad\quad mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]\}$

$\langle 2 \rangle$ DEFINE $m \triangleq [type \mapsto \text{"1b"}, acc \mapsto self, bal \mapsto b,$
$\quad m2av \mapsto 2avSent[self],$
$\quad mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]$

$\langle 2 \rangle 1.$ $bmsgs' = bmsgs \cup \{m\} \land knowsSent' = knowsSent$
BY $\langle 1 \rangle 1$ DEF $Phase1b$

$\langle 2 \rangle a.$ $\land msgsOfType(\text{"1a"})' = msgsOfType(\text{"1a"})$
$\land 1bmsgs' = 1bmsgs \cup \{1bRestrict(m)\}$
$\land 1cmsgs' = 1cmsgs$
$\land 2amsgs' = 2amsgs$
$\land acceptorMsgsOfType(\text{"2b"})' = acceptorMsgsOfType(\text{"2b"})$
BY $\langle 2 \rangle 1$ DEF $msgsOfType$, $1bmsgs$, $acceptorMsgsOfType$, $KnowsSafeAt$, $1cmsgs$, $2amsgs$

$\langle 2 \rangle$.QED
BY $\langle 2 \rangle a$ DEF $msgs$, $1bRestrict$

$\langle 1 \rangle 2.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$, $Phase2av(self, b)$
PROVE $\lor msgs' = msgs$

$\vee \exists v \in Value:$
$\wedge [type \mapsto \text{“1c”}, bal \mapsto b, val \mapsto v] \in msgs$
$\wedge msgs' = msgs \cup \{[type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v]\}$

$\langle 2 \rangle 1.$ PICK $m \in sentMsgs(\text{“1c”}, b):$
$\wedge KnowsSafeAt(self, b, m.val)$
$\wedge bmsgs' = bmsgs \cup$
$\{[type \mapsto \text{“2av”}, bal \mapsto b, val \mapsto m.val, acc \mapsto self]\}$
BY $\langle 1 \rangle 2$ DEF $Phase2av$

$\langle 2 \rangle 2.$ $m = [type \mapsto \text{“1c”}, bal \mapsto b, val \mapsto m.val]$
BY $BMessageLemma$ DEF $sentMsgs$, $TypeOK$, $1cMessage$

$\langle 2 \rangle$ DEFINE $ma \triangleq [type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto m.val]$
$mb \triangleq [type \mapsto \text{“2av”}, bal \mapsto b, val \mapsto m.val, acc \mapsto self]$

$\langle 2 \rangle 3.$ SUFFICES ASSUME $msgs' \neq msgs$
PROVE $\wedge m \in msgs$
$\wedge msgs' = msgs \cup \{ma\}$
BY $\langle 2 \rangle 2$, $BMessageLemma$ DEF $sentMsgs$, $TypeOK$, $1cMessage$

$\langle 2 \rangle 4.$ $m \in msgs$
BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ DEF $sentMsgs$, $1cmsgs$, $msgsOfType$, $msgs$

$\langle 2 \rangle 5.$ $msgs' = msgs \cup \{ma\}$

$\langle 3 \rangle 1.$ $knowsSent' = knowsSent$
BY $\langle 1 \rangle 2$ DEF $Phase2av$

$\langle 3 \rangle 2.$ $\wedge msgsOfType(\text{“1a”})' = msgsOfType(\text{“1a”})$
$\wedge 1bmsgs' = 1bmsgs$
$\wedge 1cmsgs' = 1cmsgs$
$\wedge acceptorMsgsOfType(\text{“2b”})' = acceptorMsgsOfType(\text{“2b”})$
BY $\langle 2 \rangle 1$, $\langle 3 \rangle 1$ DEF $msgsOfType$, $1bmsgs$, $1bRestrict$, $acceptorMsgsOfType$, $KnowsSafeAt$, $1cmsgs$

$\langle 3 \rangle$.QED
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$, $\langle 2 \rangle 3$ DEF $msgs$, $2amsgs$, $msgsOfType$, $acceptorMsgsOfType$

$\langle 2 \rangle 6.$ QED
BY $\langle 2 \rangle 4$, $\langle 2 \rangle 5$

$\langle 1 \rangle 3.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$, $Phase2b(self, b)$
PROVE $\exists v \in Value:$
$\wedge \exists Q \in ByzQuorum:$
$\forall a \in Q:$
$\exists m \in sentMsgs(\text{“2av”}, b): \wedge m.val = v$
$\wedge m.acc = a$
$\wedge msgs' = msgs \cup$
$\{[type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$
$\wedge bmsgs' = bmsgs \cup$
$\{[type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$
$\wedge maxVVal' = [maxVVal$ EXCEPT $![self] = v]$

$\langle 2 \rangle 1.$ PICK $v \in Value:$
$\wedge \exists Q \in ByzQuorum:$
$\forall a \in Q:$

```
                  ∃ m ∈ sentMsgs("2av", b) : ∧ m.val = v
                                             ∧ m.acc = a
              ∧ bmsgs' = bmsgs ∪
                     {[type ↦ "2b", acc ↦ self, bal ↦ b, val ↦ v]}
              ∧ maxVVal' = [maxVVal EXCEPT ![self] = v]
              ∧ knowsSent' = knowsSent
    BY ⟨1⟩3, Zenon DEF Phase2b
  ⟨2⟩ DEFINE bm ≜ [type ↦ "2b", acc ↦ self, bal ↦ b, val ↦ v]
  ⟨2⟩2. ∧ msgsOfType("1a")' = msgsOfType("1a")
        ∧ 1bmsgs' = 1bmsgs
        ∧ 1cmsgs' = 1cmsgs
        ∧ 2amsgs' = 2amsgs
        ∧ acceptorMsgsOfType("2b")' = acceptorMsgsOfType("2b") ∪ {bm}
    BY ⟨2⟩1 DEF msgsOfType, 1bmsgs, 1bRestrict, 1cmsgs, KnowsSafeAt, 2amsgs, acceptorMsgsOfType
  ⟨2⟩4. msgs' = msgs ∪ {bm}
    BY ⟨2⟩2 DEF msgs
  ⟨2⟩.QED
    BY ⟨2⟩1, ⟨2⟩4, Zenon

⟨1⟩4. ASSUME NEW self ∈ Acceptor, NEW b ∈ Ballot, LearnsSent(self, b)
      PROVE  ∃ S ∈ SUBSET {m ∈ msgsOfType("1c") : m.bal = b} : msgs' = msgs ∪ S
  ⟨2⟩1. ∧ msgsOfType("1a")' = msgsOfType("1a")
        ∧ 1bmsgs' = 1bmsgs
        ∧ 2amsgs' = 2amsgs
        ∧ acceptorMsgsOfType("2b")' = acceptorMsgsOfType("2b")
    BY ⟨1⟩4 DEF LearnsSent, msgsOfType, 1bmsgs, 1bRestrict, 2amsgs, acceptorMsgsOfType
  ⟨2⟩. ∧ 1cmsgs ⊆ 1cmsgs'
       ∧ 1cmsgs' \ 1cmsgs ∈ SUBSET {m ∈ msgsOfType("1c") : m.bal = b}
    ⟨3⟩1. bmsgs' = bmsgs
      BY ⟨1⟩4 DEF LearnsSent
    ⟨3⟩2. PICK S ∈ SUBSET sentMsgs("1b", b) :
              knowsSent' = [knowsSent EXCEPT ![self] = knowsSent[self] ∪ S]
      BY ⟨1⟩4, Zenon DEF LearnsSent
    ⟨3⟩3. ASSUME NEW m ∈ 1cmsgs
          PROVE  m ∈ 1cmsgs'
      BY ⟨3⟩1, ⟨3⟩2 DEF TypeOK, KnowsSafeAt, 1cmsgs, msgsOfType
    ⟨3⟩4. ASSUME NEW m ∈ 1cmsgs', m ∉ 1cmsgs
          PROVE  m ∈ msgsOfType("1c") ∧ m.bal = b
      ⟨4⟩1. m ∈ msgsOfType("1c")
        BY ⟨3⟩1 DEF 1cmsgs, msgsOfType
      ⟨4⟩2. PICK a ∈ Acceptor : KnowsSafeAt(a, m.bal, m.val)'
        BY DEF 1cmsgs
      ⟨4⟩3. ¬KnowsSafeAt(a, m.bal, m.val)
        BY ⟨3⟩4, ⟨4⟩1 DEF 1cmsgs
      ⟨4⟩4. ∀ aa ∈ Acceptor, bb ∈ Ballot :
```

$\qquad\qquad \forall\, mm \in KSet(aa,\, bb)' :$

$\qquad\qquad\quad mm \notin KSet(aa,\, bb) \Rightarrow bb = b$

$\qquad\quad$ BY $\langle 1\rangle 4$, $\langle 3\rangle 2$ DEF $TypeOK$, $LearnsSent$, $TypeOK$, $sentMsgs$, $KSet$

$\qquad \langle 4\rangle 5.\ m.bal \in Ballot$

$\qquad\quad$ BY $\langle 4\rangle 1$, $BMessageLemma$ DEF $1cMessage$, $msgsOfType$, $TypeOK$

$\qquad \langle 4\rangle 6.$CASE $KS1(KSet(a,\, m.bal)') \land \neg KS1(KSet(a,\, m.bal))$

$\qquad\quad$ BY $\langle 4\rangle 6$, $\langle 4\rangle 1$, $\langle 4\rangle 4$, $\langle 4\rangle 5$ DEF $KS1$

$\qquad \langle 4\rangle 7.$CASE $KS2(m.val,\, m.bal,\, KSet(a,\, m.bal)') \land \neg KS2(m.val,\, m.bal,\, KSet(a,\, m.bal))$

$\qquad\quad$ BY $\langle 4\rangle 7$, $\langle 4\rangle 1$, $\langle 4\rangle 4$, $\langle 4\rangle 5$ DEF $KS2$

$\qquad \langle 4\rangle$ QED

$\qquad\quad$ BY $\langle 4\rangle 6$, $\langle 4\rangle 7$, $\langle 4\rangle 2$, $\langle 4\rangle 3$, $KnowsSafeAtDef$

$\quad \langle 3\rangle 5.$ QED

$\quad\quad$ BY $\langle 3\rangle 3$, $\langle 3\rangle 4$

$\langle 2\rangle.$WITNESS $1cmsgs' \setminus 1cmsgs \in$ SUBSET $\{m \in msgsOfType(\text{“1c”}) : m.bal = b\}$

$\langle 2\rangle.$QED

$\quad$ BY $\langle 2\rangle 1$ DEF $msgs$

$\langle 1\rangle 5.$ ASSUME NEW $self \in Ballot$, $Phase1a(self)$

$\qquad$ PROVE $msgs' = msgs \cup \{[type \mapsto \text{“1a”},\, bal \mapsto self]\}$

BY $\langle 1\rangle 5$ DEF $Phase1a$, $msgs$, $msgsOfType$, $1bmsgs$, $1bRestrict$, $1cmsgs$, $KnowsSafeAt$, $2amsgs$, $acceptorMsgsOfType$

$\langle 1\rangle 6.$ ASSUME NEW $self \in Ballot$, $Phase1c(self)$

$\qquad$ PROVE $\exists\, S \in$ SUBSET $[type : \{\text{“1c”}\},\, bal : \{self\},\, val : Value] :$

$\qquad\qquad \land \forall\, m \in S :$

$\qquad\qquad\qquad \exists\, a \in Acceptor : KnowsSafeAt(a,\, m.bal,\, m.val)$

$\qquad\qquad \land\, msgs' = msgs \cup S$

$\langle 2\rangle 1.$ PICK $S \in$ SUBSET $[type : \{\text{“1c”}\},\, bal : \{self\},\, val : Value] :$

$\qquad\qquad \land\, bmsgs' = bmsgs \cup S$

$\qquad\qquad \land\, knowsSent' = knowsSent$

$\quad$ BY $\langle 1\rangle 6$ DEF $Phase1c$

$\langle 2\rangle$ DEFINE $SS \triangleq \{m \in S : \exists\, a \in Acceptor : KnowsSafeAt(a,\, m.bal,\, m.val)\}$

$\langle 2\rangle$ SUFFICES $msgs' = msgs \cup SS$

$\quad$ BY $\langle 2\rangle 1$, $Zenon$

$\langle 2\rangle 2.\ \land\, msgsOfType(\text{“1a”})' = msgsOfType(\text{“1a”})$

$\qquad \land\, 1bmsgs' = 1bmsgs$

$\qquad \land\, 1cmsgs' = 1cmsgs \cup SS$

$\qquad \land\, 2amsgs' = 2amsgs$

$\qquad \land\, acceptorMsgsOfType(\text{“2b”})' = acceptorMsgsOfType(\text{“2b”})$

BY $\langle 2\rangle 1$ DEF $msgsOfType$, $1bmsgs$, $1bRestrict$, $1cmsgs$, $KnowsSafeAt$, $2amsgs$, $acceptorMsgsOfType$

$\langle 2\rangle 3.$ QED

$\quad$ BY $\langle 2\rangle 2$ DEF $msgs$

$\langle 1\rangle 7.$ ASSUME NEW $self \in FakeAcceptor$, $FakingAcceptor(self)$

$\qquad$ PROVE $msgs' = msgs$

BY $\langle 1\rangle 7$, BQA DEF $FakingAcceptor$, $msgs$, $1bMessage$, $2avMessage$, $2bMessage$,

$\quad msgsOfType, 1cmsgs, KnowsSafeAt, 1bmsgs, 2amsgs, acceptorMsgsOfType, msgsOfType$

$\langle 1\rangle 9.$ QED

BY $\langle 1\rangle 1, \langle 1\rangle 2, \langle 1\rangle 3, \langle 1\rangle 4, \langle 1\rangle 5, \langle 1\rangle 6, \langle 1\rangle 7, Zenon$

We now come to the proof of invariance of our inductive invariant *Inv*.

THEOREM $Invariance \triangleq Spec \Rightarrow \Box Inv$

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

BY $FS\_EmptySet$ DEF $Init, Inv, TypeOK, bmsgsFinite, 1bOr2bMsgs, 1bInv1, 1bInv2,$
$maxBalInv, 2avInv1, 2avInv2, 2avInv3, accInv, knowsSentInv$

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

$\langle 2\rangle$ SUFFICES ASSUME $Inv, [Next]_{vars}$
PROVE Inv'

OBVIOUS

$\langle 2\rangle 1.$ ASSUME NEW $self \in Acceptor,$
NEW $b \in Ballot,$
$\lor Phase1b(self, b)$
$\lor Phase2av(self, b)$
$\lor Phase2b(self, b)$
$\lor LearnsSent(self, b)$
PROVE Inv'

$\langle 3\rangle 1.$ CASE $Phase1b(self, b)$

$\langle 4\rangle$ DEFINE $mb \triangleq [type \mapsto \text{"1b"}, bal \mapsto b, acc \mapsto self,$
$m2av \mapsto 2avSent[self],$
$mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]$
$mc \triangleq [type \mapsto \text{"1b"}, acc \mapsto self, bal \mapsto b,$
$mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]$

$\langle 4\rangle 1.\ msgs' = msgs \cup \{mc\}$

BY $\langle 3\rangle 1, MsgsLemma$ DEF Inv

$\langle 4\rangle 2.\ TypeOK'$

BY $\langle 3\rangle 1$ DEF $Inv, TypeOK, BMessage, 1bMessage, ByzAcceptor, Phase1b$

$\langle 4\rangle 3.\ bmsgsFinite'$

BY $\langle 3\rangle 1, FiniteMsgsLemma, Zenon$ DEF $Inv, bmsgsFinite, Phase1b$

$\langle 4\rangle 4.\ 1bInv1'$

BY $\langle 3\rangle 1, \langle 4\rangle 1, Isa$ DEF $Phase1b, 1bInv1, Inv, accInv$

$\langle 4\rangle 5.\ 1bInv2'$

BY $\langle 3\rangle 1$ DEF $Phase1b, 1bInv2, Inv, maxBalInv, TypeOK, 1bMessage, Ballot$

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

BY $\langle 3\rangle 1, BMessageLemma$ DEF $Phase1b, maxBalInv, Ballot, Inv, TypeOK,$
$1bMessage, 2avMessage, 2bMessage$

$\langle 4\rangle 7.\ 2avInv1'$

BY $\langle 3\rangle 1$ DEF $Phase1b, Inv, 2avInv1$

$\langle 4\rangle 8.\ 2avInv2'$

BY $\langle 3\rangle 1$ DEF $Phase1b, Inv, 2avInv2$

$\langle 4\rangle 9.\ 2avInv3'$
BY $\langle 3\rangle 1$, $\langle 4\rangle 1$ DEF $Phase1b$, Inv, $2avInv3$
$\langle 4\rangle 10.\ accInv'$
$\langle 5\rangle$ SUFFICES ASSUME NEW $a \in Acceptor$,
NEW $r \in 2avSent[a]$
PROVE $\land\ r.bal \leq maxBal'[a]$
$\land\ [type \mapsto \text{“1c”},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs'$
BY $\langle 3\rangle 1$, $Zenon$ DEF $accInv$, $Phase1b$
$\langle 5\rangle$ $[type \mapsto \text{“1c”},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs'$
BY $\langle 3\rangle 1$, $MsgsLemma$ DEF Inv, $accInv$
$\langle 5\rangle$ QED
BY $\langle 3\rangle 1$ DEF $Phase1b$, Inv, $Ballot$, $TypeOK$, $accInv$
$\langle 4\rangle 11.\ knowsSentInv'$
BY $\langle 3\rangle 1$ DEF $Phase1b$, Inv, $knowsSentInv$, $msgsOfType$
$\langle 4\rangle 12.$ QED
BY $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 4\rangle 4$, $\langle 4\rangle 5$, $\langle 4\rangle 6$, $\langle 4\rangle 7$, $\langle 4\rangle 8$, $\langle 4\rangle 9$, $\langle 4\rangle 10$, $\langle 4\rangle 11$ DEF Inv
$\langle 3\rangle 2.$CASE $Phase2av(self, b)$
$\langle 4\rangle 1.$ PICK $mc \in sentMsgs(\text{“1c”}, b)$:
$\land\ KnowsSafeAt(self, b, mc.val)$
$\land\ bmsgs' = bmsgs \cup \{[type \mapsto \text{“2av”},\ bal \mapsto b,\ val \mapsto mc.val,\ acc \mapsto self]\}$
$\land\ 2avSent' = [2avSent$ EXCEPT
$![self] = \{r \in 2avSent[self] : r.val \neq mc.val\} \cup \{[val \mapsto mc.val,\ bal \mapsto b]\}]$
BY $\langle 3\rangle 2$, $Zenon$ DEF $Phase2av$
$\langle 4\rangle 2.\ mc = [type \mapsto \text{“1c”},\ bal \mapsto mc.bal,\ val \mapsto mc.val]$
BY $\langle 4\rangle 1$, $BMessageLemma$ DEF $sentMsgs$, Inv, $TypeOK$, $1cMessage$
$\langle 4\rangle$ DEFINE $mb \triangleq [type \mapsto \text{“2av”},\ bal \mapsto b,\ val \mapsto mc.val,\ acc \mapsto self]$
$mmc(v) \triangleq [type \mapsto \text{“1c”},\ bal \mapsto b,\ val \mapsto v]$
$ma(v) \triangleq [type \mapsto \text{“2a”},\ bal \mapsto b,\ val \mapsto v]$
$\langle 4\rangle 3.\ \lor\ msgs' = msgs$
$\lor\ \exists\, v \in Value$:
$\land\ mmc(v) \in msgs$
$\land\ msgs' = msgs \cup \{ma(v)\}$
BY $\langle 3\rangle 2$, $MsgsLemma$, $Zenon$ DEF Inv
$\langle 4\rangle 4.\ msgs \subseteq msgs'$
BY $\langle 4\rangle 3$, $Zenon$
$\langle 4\rangle 5.\ TypeOK'$
BY $\langle 3\rangle 2$, $\langle 4\rangle 1$, $BMessageLemma$
DEF $sentMsgs$, Inv, $TypeOK$, $1cMessage$, $Phase2av$, $2avMessage$, $ByzAcceptor$, $BMessage$
$\langle 4\rangle 6.\ bmsgsFinite'$
BY $\langle 4\rangle 1$, $FiniteMsgsLemma$, $Zenon$ DEF Inv, $bmsgsFinite$

$\langle 4\rangle 7.\ 1bInv1'$

BY $\langle 3\rangle 2$, $\langle 4\rangle 1$, $\langle 4\rangle 3$, *Isa* DEF *Phase2av*, *1bInv1*, *Inv*

$\langle 4\rangle 8.\ 1bInv2'$

BY $\langle 4\rangle 1$ DEF *Inv*, *1bInv2*

$\langle 4\rangle 9.\ maxBalInv'$

BY $\langle 3\rangle 2$, $\langle 4\rangle 1$, *BMessageLemma*
DEF *Phase2av*, *maxBalInv*, *Ballot*, *Inv*, *TypeOK*, *1bMessage*, *2avMessage*, *2bMessage*

$\langle 4\rangle 10.\ 2avInv1'$

BY $\langle 3\rangle 2$, $\langle 4\rangle 1$ DEF *Phase2av*, *Inv*, *2avInv1*, *2avInv2*, *TypeOK*, *1bMessage*, *Ballot*

$\langle 4\rangle 11.\ 2avInv2'$

$\langle 5\rangle 1.$ SUFFICES ASSUME NEW $m \in bmsgs'$,
$2avInv2!(m)!1$
PROVE $\exists r \in 2avSent'[m.acc] : \land r.val = m.val$
$\land r.bal \geq m.bal$

BY DEF *2avInv2*

$\langle 5\rangle 2.$CASE $m.acc = self$

$\langle 6\rangle 1.$CASE $m = mb$

BY $\langle 4\rangle 1$, $\langle 6\rangle 1$, *Isa* DEF *Inv*, *TypeOK*, *Ballot*

$\langle 6\rangle 2.$CASE $m \neq mb$

$\langle 7\rangle 1.\ m \in bmsgs$

BY $\langle 4\rangle 1$, $\langle 6\rangle 2$

$\langle 7\rangle 2.$ PICK $r \in 2avSent[m.acc] : \land r.val = m.val$
$\land r.bal \geq m.bal$

BY $\langle 5\rangle 1$, $\langle 7\rangle 1$ DEF *Inv*, *2avInv2*

$\langle 7\rangle 3.$CASE $r.val = mc.val$

$\langle 8\rangle.$DEFINE $rr \triangleq [val \mapsto mc.val,\ bal \mapsto b]$

$\langle 8\rangle. rr \in 2avSent'[m.acc]$

BY $\langle 4\rangle 1$, $\langle 5\rangle 2$ DEF *Inv*, *TypeOK*

$\langle 8\rangle.$WITNESS $rr \in 2avSent'[m.acc]$

$\langle 8\rangle.$QED

BY $\langle 7\rangle 2$, $\langle 7\rangle 3$, $\langle 5\rangle 2$, $\langle 5\rangle 1$, $\langle 3\rangle 2$, *BMessageLemma*
DEF *Phase2av*, *Inv*, *TypeOK*, *accInv*, *Ballot*, *2avMessage*

$\langle 7\rangle 4.$CASE $r.val \neq mc.val$

BY $\langle 7\rangle 2$, $\langle 4\rangle 1$, $\langle 5\rangle 2$, $\langle 7\rangle 4$ DEF *Inv*, *TypeOK*

$\langle 7\rangle 5.$ QED

BY $\langle 7\rangle 3$, $\langle 7\rangle 4$

$\langle 6\rangle 3.$ QED

BY $\langle 6\rangle 1$, $\langle 6\rangle 2$

$\langle 5\rangle 3.$CASE $m.acc \neq self$

BY $\langle 5\rangle 3$, $\langle 5\rangle 1$, $\langle 4\rangle 1$, *BMessageLemma* DEF *Inv*, *TypeOK*, *2avInv2*, *2avMessage*

$\langle 5\rangle 4.$ QED

BY $\langle 5\rangle 2$, $\langle 5\rangle 3$

$\langle 4\rangle 12.\ 2avInv3'$

BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 4$ DEF *Inv*, *2avInv3*, *sentMsgs*, *msgs*, *1cmsgs*, *msgsOfType*

$\langle 4\rangle 13.\ accInv'$

$\langle 5\rangle 1.$ SUFFICES ASSUME NEW $a \in \mathit{Acceptor}$,
NEW $r \in 2avSent'[a]$
PROVE $\land\, r.bal \leq maxBal'[a]$
$\land\, [type \mapsto \text{“1c”},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs'$

BY Zenon DEF $accInv$

$\langle 5\rangle 2.$CASE $r \in 2avSent[a]$

BY $\langle 5\rangle 2$, $\langle 4\rangle 4$, $\langle 4\rangle 5$, $\langle 3\rangle 2$ DEF $Phase2av$, Inv, $TypeOK$, $accInv$, $Ballot$

$\langle 5\rangle 3.$CASE $r \notin 2avSent[a]$

BY $\langle 5\rangle 3$, $\langle 3\rangle 2$, $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 4\rangle 4$
DEF $Phase2av$, Inv, $TypeOK$, $sentMsgs$, $msgsOfType$, $msgs$, $1cmsgs$, $Ballot$

$\langle 5\rangle 4.$ QED

BY $\langle 5\rangle 2$, $\langle 5\rangle 3$

$\langle 4\rangle 14.$ $knowsSentInv'$

BY $\langle 3\rangle 2$, $\langle 4\rangle 1$ DEF $Phase2av$, Inv, $knowsSentInv$, $msgsOfType$

$\langle 4\rangle 15.$ QED

BY $\langle 4\rangle 5$, $\langle 4\rangle 6$, $\langle 4\rangle 7$, $\langle 4\rangle 8$, $\langle 4\rangle 9$, $\langle 4\rangle 10$, $\langle 4\rangle 11$, $\langle 4\rangle 12$, $\langle 4\rangle 13$, $\langle 4\rangle 14$ DEF Inv

$\langle 3\rangle 3.$CASE $Phase2b(self, b)$

$\langle 4\rangle 1.$ PICK $v \in Value$:
$\land\ \exists\, Q \in ByzQuorum :$
$\forall\, a \in Q :$
$\exists\, m \in sentMsgs(\text{“2av”}, b) : \land\, m.val = v$
$\land\, m.acc = a$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“2b”},\ acc \mapsto self,\ bal \mapsto b,\ val \mapsto v]\}$
$\land\ bmsgs' = (bmsgs \cup \{[type \mapsto \text{“2b”},\ acc \mapsto self,\ bal \mapsto b,\ val \mapsto v]\})$
$\land\ maxVVal' = [maxVVal$ EXCEPT $![self] = v]$

BY $\langle 3\rangle 3$, $MsgsLemma$ DEF Inv

$\langle 4\rangle$ DEFINE $mb \triangleq [type \mapsto \text{“2b”},\ acc \mapsto self,\ bal \mapsto b,\ val \mapsto v]$

$\langle 4\rangle 2.$ $TypeOK'$

BY $\langle 3\rangle 3$, $\langle 4\rangle 1$ DEF $Phase2b$, Inv, $TypeOK$, $BMessage$, $2bMessage$, $ByzAcceptor$

$\langle 4\rangle 3.$ $bmsgsFinite'$

BY $\langle 4\rangle 1$, $FiniteMsgsLemma$, $Zenon$ DEF Inv, $bmsgsFinite$

$\langle 4\rangle 4.$ $1bInv1'$

BY $\langle 4\rangle 1$, Isa DEF Inv, $1bInv1$

$\langle 4\rangle 5.$ $1bInv2'$

BY $\langle 4\rangle 1$ DEF Inv, $1bInv2$

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

BY $\langle 3\rangle 3$, $\langle 4\rangle 1$, $\langle 4\rangle 2$, $BMessageLemma$
DEF $Phase2b$, Inv, $maxBalInv$, $TypeOK$, $Ballot$, $1bMessage$, $2avMessage$, $2bMessage$

$\langle 4\rangle 7.$ $2avInv1'$

BY $\langle 4\rangle 1$ DEF Inv, $2avInv1$

$\langle 4\rangle 8.$ $2avInv2'$

BY $\langle 3\rangle 3$, $\langle 4\rangle 1$ DEF $Phase2b$, Inv, $TypeOK$, $2avInv2$

$\langle 4\rangle 9.\ 2avInv3'$
  BY $\langle 4\rangle 1$ DEF Inv, $2avInv3$
$\langle 4\rangle 10.\ accInv'$
  $\langle 5\rangle$ SUFFICES ASSUME NEW $a \in Acceptor$,
    NEW $r \in 2avSent[a]$
    PROVE $\land\ r.bal \leq maxBal'[a]$
      $\land\ [type \mapsto \text{“1c”},\ bal \mapsto r.bal,\ val \mapsto r.val]$
        $\in msgs'$
    BY $\langle 3\rangle 3$, $Zenon$ DEF $accInv$, $Phase2b$
  $\langle 5\rangle$ $[type \mapsto \text{“1c”},\ bal \mapsto r.bal,\ val \mapsto r.val] \in msgs'$
    BY $\langle 3\rangle 3$, $MsgsLemma$ DEF Inv, $accInv$
  $\langle 5\rangle$ QED
    BY $\langle 3\rangle 3$ DEF $Phase2b$, Inv, $Ballot$, $TypeOK$, $accInv$
$\langle 4\rangle 11.\ knowsSentInv'$
  BY $\langle 3\rangle 3$, $\langle 4\rangle 1$ DEF $Phase2b$, Inv, $knowsSentInv$, $msgsOfType$
$\langle 4\rangle 12.$ QED
  BY $\langle 4\rangle 2$, $\langle 4\rangle 3$, $\langle 4\rangle 4$, $\langle 4\rangle 5$, $\langle 4\rangle 6$, $\langle 4\rangle 7$, $\langle 4\rangle 8$, $\langle 4\rangle 9$, $\langle 4\rangle 10$, $\langle 4\rangle 11$ DEF Inv
$\langle 3\rangle 4.$CASE $LearnsSent(self,\ b)$
$\langle 4\rangle 1.$ PICK $MS :\ \land\ MS \subseteq \{m \in msgsOfType(\text{“1c”}) : m.bal = b\}$
    $\land\ msgs' = msgs \cup MS$
  BY $\langle 3\rangle 4$, $MsgsLemma$, $Zenon$ DEF Inv
$\langle 4\rangle 2.$ PICK S :
    $\land\ \ S \subseteq sentMsgs(\text{“1b”},\ b)$
    $\land\ \ knowsSent' =$
      $[knowsSent$ EXCEPT $![self] = knowsSent[self] \cup S]$
  BY $\langle 3\rangle 4$, $Zenon$ DEF $LearnsSent$
$\langle 4\rangle 3.\ TypeOK'$
  BY $\langle 3\rangle 4$, $\langle 4\rangle 2$, $BMessageLemma$ DEF Inv, $TypeOK$, $sentMsgs$, $LearnsSent$
$\langle 4\rangle 4.\ bmsgsFinite'$
  BY $\langle 3\rangle 4$ DEF $LearnsSent$, Inv, $bmsgsFinite$, $1bOr2bMsgs$
$\langle 4\rangle 5.\ 1bInv1'$
  BY $\langle 3\rangle 4$, $\langle 4\rangle 1$, $Zenon$ DEF $LearnsSent$, Inv, $1bInv1$
$\langle 4\rangle 6.\ 1bInv2'$
  BY $\langle 3\rangle 4$ DEF $LearnsSent$, Inv, $1bInv2$
$\langle 4\rangle 7.\ maxBalInv'$
  BY $\langle 3\rangle 4$ DEF $LearnsSent$, Inv, $maxBalInv$
$\langle 4\rangle 8.\ 2avInv1'$
  BY $\langle 3\rangle 4$ DEF $LearnsSent$, Inv, $2avInv1$
$\langle 4\rangle 9.\ 2avInv2'$
  BY $\langle 3\rangle 4$ DEF $LearnsSent$, Inv, $2avInv2$
$\langle 4\rangle 10.\ 2avInv3'$
  BY $\langle 3\rangle 4$, $\langle 4\rangle 1$ DEF $LearnsSent$, Inv, $2avInv3$
$\langle 4\rangle 11.\ accInv'$
  BY $\langle 3\rangle 4$, $\langle 4\rangle 1$, $Zenon$ DEF $LearnsSent$, Inv, $accInv$
$\langle 4\rangle 12.\ knowsSentInv'$

BY ⟨3⟩4, ⟨4⟩2 DEF *LearnsSent*, *Inv*, *TypeOK*, *knowsSentInv*, *sentMsgs*, *msgsOfType*

⟨4⟩13. QED

BY ⟨4⟩3, ⟨4⟩4, ⟨4⟩5, ⟨4⟩6, ⟨4⟩7, ⟨4⟩8, ⟨4⟩9, ⟨4⟩10, ⟨4⟩11, ⟨4⟩12 DEF *Inv*

⟨3⟩5. QED

BY ⟨2⟩1, ⟨3⟩1, ⟨3⟩2, ⟨3⟩3, ⟨3⟩4

⟨2⟩2. ASSUME NEW $self \in Ballot$,

$\lor\ Phase1a(self)$

$\lor\ Phase1c(self)$

PROVE Inv'

⟨3⟩1.CASE $Phase1a(self)$

⟨4⟩ DEFINE $ma \triangleq [type \mapsto \text{"1a"},\ bal \mapsto self]$

⟨4⟩1. $msgs' = msgs \cup \{ma\}$

BY ⟨3⟩1, *MsgsLemma* DEF *Inv*

⟨4⟩2. $TypeOK'$

BY ⟨3⟩1 DEF *Phase1a*, *Inv*, *TypeOK*, *BMessage*, *1aMessage*

⟨4⟩3. $bmsgsFinite'$

BY ⟨3⟩1, *FiniteMsgsLemma*, *Zenon* DEF *Inv*, *bmsgsFinite*, *Phase1a*

⟨4⟩4. $1bInv1'$

BY ⟨3⟩1, ⟨4⟩1, *Isa* DEF *Phase1a*, *Inv*, *1bInv1*

⟨4⟩5. $1bInv2'$

BY ⟨3⟩1 DEF *Phase1a*, *Inv*, *1bInv2*

⟨4⟩6. $maxBalInv'$

BY ⟨3⟩1 DEF *Phase1a*, *Inv*, *maxBalInv*

⟨4⟩7. $2avInv1'$

BY ⟨3⟩1 DEF *Phase1a*, *Inv*, *2avInv1*

⟨4⟩8. $2avInv2'$

BY ⟨3⟩1 DEF *Phase1a*, *Inv*, *2avInv2*

⟨4⟩9. $2avInv3'$

BY ⟨3⟩1, ⟨4⟩1 DEF *Phase1a*, *Inv*, *2avInv3*

⟨4⟩10. $accInv'$

BY ⟨3⟩1, ⟨4⟩1, *Zenon* DEF *Phase1a*, *Inv*, *accInv*

⟨4⟩11. $knowsSentInv'$

BY ⟨3⟩1 DEF *Inv*, *knowsSentInv*, *msgsOfType*, *Phase1a*

⟨4⟩12. QED

BY ⟨4⟩2, ⟨4⟩3, ⟨4⟩4, ⟨4⟩5, ⟨4⟩6, ⟨4⟩7, ⟨4⟩8, ⟨4⟩9, ⟨4⟩10, ⟨4⟩11 DEF *Inv*

⟨3⟩2.CASE $Phase1c(self)$

⟨4⟩1. PICK $S : \land\ S \in \text{SUBSET}\ [type : \{\text{"1c"}\},\ bal : \{self\},\ val : Value]$

$\land\ bmsgs' = bmsgs \cup S$

BY ⟨3⟩2 DEF *Phase1c*

⟨4⟩2. PICK MS :

$\land\ MS \in \text{SUBSET}\ [type : \{\text{"1c"}\},\ bal : \{self\},\ val : Value]$

$\land\ \forall\, m \in MS :$

$\exists\, a \in Acceptor : KnowsSafeAt(a,\ m.bal,\ m.val)$

$\land\ msgs' = msgs \cup MS$

BY ⟨3⟩2, *MsgsLemma* DEF *Inv*

$\langle 4\rangle 3.$ *TypeOK'*
  BY $\langle 3\rangle 2$, $\langle 4\rangle 1$ DEF *Phase1c*, *Inv*, *TypeOK*, *BMessage*, *1cMessage*
$\langle 4\rangle 4.$ *bmsgsFinite'*
  BY $\langle 4\rangle 1$ DEF *Inv*, *bmsgsFinite*, *1bOr2bMsgs*
$\langle 4\rangle 5.$ *1bInv1'*
  BY $\langle 3\rangle 2$, $\langle 4\rangle 2$, *Zenon* DEF *Phase1c*, *Inv*, *1bInv1*
$\langle 4\rangle 6.$ *1bInv2'*
  BY $\langle 4\rangle 1$ DEF *Inv*, *1bInv2*
$\langle 4\rangle 7.$ *maxBalInv'*
  BY $\langle 3\rangle 2$ DEF *Phase1c*, *Inv*, *maxBalInv*
$\langle 4\rangle 8.$ *2avInv1'*
  BY $\langle 4\rangle 1$ DEF *Inv*, *2avInv1*
$\langle 4\rangle 9.$ *2avInv2'*
  BY $\langle 3\rangle 2$ DEF *Phase1c*, *Inv*, *2avInv2*
$\langle 4\rangle 10.$ *2avInv3'*
  BY $\langle 3\rangle 2$, $\langle 4\rangle 2$ DEF *Phase1c*, *Inv*, *2avInv3*
$\langle 4\rangle 11.$ *accInv'*
  BY $\langle 3\rangle 2$, $\langle 4\rangle 2$, *Zenon* DEF *Phase1c*, *Inv*, *accInv*
$\langle 4\rangle 12.$ *knowsSentInv'*
  BY $\langle 3\rangle 2$ DEF *Inv*, *knowsSentInv*, *msgsOfType*, *Phase1c*
$\langle 4\rangle 13.$ QED
  BY $\langle 4\rangle 3$, $\langle 4\rangle 4$, $\langle 4\rangle 5$, $\langle 4\rangle 6$, $\langle 4\rangle 7$, $\langle 4\rangle 8$, $\langle 4\rangle 9$, $\langle 4\rangle 10$, $\langle 4\rangle 11$, $\langle 4\rangle 12$ DEF *Inv*
$\langle 3\rangle 3.$ QED
  BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 2\rangle 2$
$\langle 2\rangle 3.$ ASSUME NEW $self \in FakeAcceptor$,
  $FakingAcceptor(self)$
  PROVE Inv'
$\langle 3\rangle 1.$ PICK $m \in 1bMessage \cup 2avMessage \cup 2bMessage$:
  $\land\ m.acc \notin Acceptor$
  $\land\ bmsgs' = bmsgs \cup \{m\}$
  BY $\langle 2\rangle 3$, *BQA* DEF *FakingAcceptor*
$\langle 3\rangle 2.$ $msgs' = msgs$
  BY $\langle 2\rangle 3$, *MsgsLemma* DEF *Inv*
$\langle 3\rangle 3.$ *TypeOK'*
  BY $\langle 2\rangle 3$, $\langle 3\rangle 1$ DEF *Inv*, *TypeOK*, *BMessage*, *FakingAcceptor*
$\langle 3\rangle 4.$ *bmsgsFinite'*
  BY $\langle 3\rangle 1$, *FiniteMsgsLemma* DEF *Inv*, *TypeOK*
$\langle 3\rangle 5.$ *1bInv1'*
  BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, *Zenon* DEF *Inv*, *1bInv1*
$\langle 3\rangle 6.$ *1bInv2'*
  BY $\langle 3\rangle 1$ DEF *Inv*, *1bInv2*
$\langle 3\rangle 7.$ *maxBalInv'*
  BY $\langle 2\rangle 3$, $\langle 3\rangle 1$ DEF *Inv*, *maxBalInv*, *FakingAcceptor*
$\langle 3\rangle 8.$ *2avInv1'*
  BY $\langle 3\rangle 1$ DEF *Inv*, *2avInv1*

$\langle 3 \rangle 9.\ 2avInv2'$
  BY $\langle 2 \rangle 3$, $\langle 3 \rangle 1$ DEF Inv, $2avInv2$, $FakingAcceptor$
$\langle 3 \rangle 10.\ 2avInv3'$
  BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$ DEF Inv, $2avInv3$
$\langle 3 \rangle 11.\ accInv'$
  BY $\langle 2 \rangle 3$, $\langle 3 \rangle 2$, $Zenon$ DEF Inv, $accInv$, $FakingAcceptor$
$\langle 3 \rangle 12.\ knowsSentInv'$
  BY $\langle 2 \rangle 3$, $\langle 3 \rangle 1$ DEF Inv, $knowsSentInv$, $msgsOfType$, $FakingAcceptor$
$\langle 3 \rangle 13.$ QED
  BY $\langle 3 \rangle 3$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $\langle 3 \rangle 6$, $\langle 3 \rangle 7$, $\langle 3 \rangle 8$, $\langle 3 \rangle 9$, $\langle 3 \rangle 10$, $\langle 3 \rangle 11$, $\langle 3 \rangle 12$ DEF Inv
$\langle 2 \rangle 4.$ ASSUME UNCHANGED $vars$
  PROVE Inv'
$\langle 3 \rangle$ USE UNCHANGED $vars$ DEF Inv, $vars$
$\langle 3 \rangle$ $msgs = msgs'$
  BY DEF $msgs$, $msgsOfType$, $1bmsgs$, $1bRestrict$, $acceptorMsgsOfType$, $1cmsgs$, $KnowsSafeAt$, $2amsgs$
$\langle 3 \rangle$ QED
  BY DEF $TypeOK$, $bmsgsFinite$, $1bOr2bMsgs$, $1bInv1$, $1bInv2$, $maxBalInv$, $2avInv1$, $2avInv2$, $2avInv3$, $accInv$, $knowsSentInv$, $msgsOfType$
$\langle 2 \rangle 5.$ QED
  BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $NextDef$

$\langle 1 \rangle 3.$ QED
  BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, PTL DEF $Spec$

We next use the invariance of *Inv* to prove that algorithm *BPCon* implements algorithm *PCon* under the refinement mapping defined by the INSTANCE statement above.

THEOREM $Spec \Rightarrow P!Spec$
$\langle 1 \rangle 1.\ Init \Rightarrow P!Init$
$\langle 2 \rangle$.HAVE $Init$
$\langle 2 \rangle 1.\ MaxBallot(\{\}) = -1$
  BY $MaxBallotProp$, $FS\_EmptySet$
$\langle 2 \rangle 2.\ P!Init!1 \wedge P!Init!2 \wedge P!Init!3$
  BY $\langle 2 \rangle 1$ DEF $Init$, $PmaxBal$, $1bOr2bMsgs$, $None$, $P!None$
$\langle 2 \rangle 3.\ msgs = \{\}$
  BY BQA DEF $Init$, $msgsOfType$, $acceptorMsgsOfType$, $1bmsgs$, $1cmsgs$, $2amsgs$, $Quorum$, $msgs$
$\langle 2 \rangle 4.$ QED
  BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$ DEF $P!Init$

$\langle 1 \rangle 2.\ Inv \wedge Inv' \wedge [Next]_{vars} \Rightarrow [P!Next]_{P!vars}$
$\langle 2 \rangle$ $InvP \triangleq Inv'$
$\langle 2 \rangle$ SUFFICES ASSUME Inv, $InvP$, $Next$
  PROVE $P!TLANext \vee P!vars' = P!vars$
$\langle 3 \rangle$ UNCHANGED $vars \Rightarrow$ UNCHANGED $P!vars$
  BY DEF $vars$, $P!vars$, $PmaxBal$, $1bOr2bMsgs$, $msgs$, $msgsOfType$, $acceptorMsgsOfType$,

$1bmsgs$, $2amsgs$, $1cmsgs$, $KnowsSafeAt$
⟨3⟩ QED
BY $PNextDef$ DEF Inv, $P!ProcSet$, $P!Init$, $Ballot$, $P!Ballot$
⟨2⟩ HIDE DEF $InvP$
⟨2⟩2. $\forall a \in Acceptor : PmaxBal[a] \in Ballot \cup \{-1\}$
BY $PMaxBalLemma3$, $MaxBallotProp$ DEF Inv, $PmaxBal$, $1bOr2bMsgs$
⟨2⟩3. ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$,
$Phase1b(self, b)$
PROVE $P!TLANext \lor P!vars' = P!vars$
⟨3⟩1. $msgs' = msgs \cup \{[type \mapsto \text{"1b"}, acc \mapsto self, bal \mapsto b,$
$mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]\}$
BY ⟨2⟩3, $MsgsLemma$ DEF Inv
⟨3⟩2. $P!sentMsgs(\text{"1a"}, b) \neq \{\}$
BY ⟨2⟩3 DEF $Phase1b$, $sentMsgs$, $msgsOfType$, $msgs$, $P!sentMsgs$
⟨3⟩3. UNCHANGED $\langle maxVBal, maxVVal \rangle$
BY ⟨2⟩3 DEF $Phase1b$
⟨3⟩4. $b > PmaxBal[self]$
BY ⟨2⟩2, ⟨2⟩3, $PmaxBalLemma4$ DEF $Phase1b$, Inv, $TypeOK$, $Ballot$
⟨3⟩5. $PmaxBal' = [PmaxBal$ EXCEPT $![self] = b]$
⟨4⟩ DEFINE $m \triangleq [type \mapsto \text{"1b"}, bal \mapsto b, acc \mapsto self,$
$m2av \mapsto 2avSent[self],$
$mbal \mapsto maxVBal[self], mval \mapsto maxVVal[self]]$
$mA(a) \triangleq \{ma \in bmsgs : \land\ ma.type \in \{\text{"1b"}, \text{"2b"}\}$
$\land\ ma.acc = a\}$
$S(a) \triangleq \{ma.bal : ma \in mA(a)\}$
⟨4⟩1. $bmsgs' = bmsgs \cup \{m\}$
BY ⟨2⟩3 DEF $Phase1b$
⟨4⟩2. $mA(self)' = mA(self) \cup \{m\}$
BY ⟨4⟩1
⟨4⟩3. $\land\ PmaxBal = [a \in Acceptor \mapsto MaxBallot(S(a))]$
$\land\ PmaxBal' = [a \in Acceptor \mapsto MaxBallot(S(a))']$
BY DEF $PmaxBal$, $1bOr2bMsgs$
⟨4⟩ HIDE DEF mA
⟨4⟩4. $S(self)' = S(self) \cup \{b\}$
BY ⟨4⟩2, Isa
⟨4⟩5. $MaxBallot(S(self) \cup \{b\}) = b$
⟨5⟩ DEFINE $SS \triangleq S(self) \cup \{b\}$
⟨5⟩1. $IsFiniteSet(S(self))$
⟨6⟩.$IsFiniteSet(mA(self))$
BY FS_Subset DEF Inv, $bmsgsFinite$, mA, $1bOr2bMsgs$
⟨6⟩.QED
BY FS_Image, Isa
⟨5⟩2. $IsFiniteSet(SS)$
BY ⟨5⟩1, $FS\_AddElement$
⟨5⟩3. $S(self) \subseteq Ballot \cup \{-1\}$

BY $BMessageLemma$ DEF mA, Inv, $TypeOK$, $1bMessage$, $2bMessage$

$\langle 5\rangle 4.\ \forall x \in SS : b \geq x$

BY $\langle 3\rangle 4$, $\langle 4\rangle 3$, $\langle 5\rangle 1$, $\langle 5\rangle 3$, $MaxBallotProp$, $Z3T(10)$ DEF $Ballot$

$\langle 5\rangle 5.$ QED

BY $\langle 5\rangle 2$, $\langle 5\rangle 3$, $\langle 5\rangle 4$, $MaxBallotLemma1$

$\langle 4\rangle 6.\ \forall a \in Acceptor : a \neq self \Rightarrow S(a)' = S(a)$

BY $\langle 4\rangle 1$ DEF mA

$\langle 4\rangle 7.$ QED

BY $\langle 4\rangle 3$, $\langle 4\rangle 4$, $\langle 4\rangle 5$, $\langle 4\rangle 6$, $Zenon$ DEF $PmaxBal$, $1bOr2bMsgs$

$\langle 3\rangle 6.$ QED

BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$, $\langle 3\rangle 4$, $\langle 3\rangle 5$, $Zenon$ DEF $P!TLANext$, $P!Ballot$, $Ballot$, $P!Phase1b$

$\langle 2\rangle 4.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$,
$Phase2av(self, b)$
PROVE $P!TLANext \vee P!vars' = P!vars$

$\langle 3\rangle 1.\ PmaxBal' = PmaxBal$

$\langle 4\rangle$ DEFINE $mm(m) \triangleq [type \mapsto \text{“2av”}, bal \mapsto b, val \mapsto m.val, acc \mapsto self]$

$\langle 4\rangle 1.$ PICK $m : bmsgs' = bmsgs \cup \{mm(m)\}$

BY $\langle 2\rangle 4$ DEF $Phase2av$

$\langle 4\rangle 2.\ mm(m).type = \text{“2av”}$

OBVIOUS

$\langle 4\rangle$ QED

BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $PmaxBalLemma1$, $Zenon$

$\langle 3\rangle 2.$CASE $msgs' = msgs$

BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 2\rangle 4$ DEF $Phase2av$, $P!vars$

$\langle 3\rangle 3.$CASE $\wedge\ msgs' \neq msgs$
$\wedge\ \exists v \in Value :$
$\wedge\ [type \mapsto \text{“1c”}, bal \mapsto b, val \mapsto v] \in msgs$
$\wedge\ msgs' = msgs \cup \{[type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v]\}$

$\langle 4\rangle 1.$ PICK $v \in Value :$
$\wedge\ [type \mapsto \text{“1c”}, bal \mapsto b, val \mapsto v] \in msgs$
$\wedge\ msgs' = msgs \cup \{[type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v]\}$

BY $\langle 3\rangle 3$

$\langle 4\rangle 2.\ P!sentMsgs(\text{“2a”}, b) = \{\}$

$\langle 5\rangle 1.$ SUFFICES ASSUME NEW $m \in P!sentMsgs(\text{“2a”}, b)$
PROVE $m = [type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v]$

BY $\langle 3\rangle 3$, $\langle 4\rangle 1$ DEF $P!sentMsgs$

$\langle 5\rangle 2.\ \wedge\ m \in 2amsgs$
$\wedge\ m.type = \text{“2a”}$
$\wedge\ m.bal = b$

BY $MsgsTypeLemma$ DEF $P!sentMsgs$

$\langle 5\rangle 3.$ PICK $Q \in Quorum :$
$\forall a \in Q :$
$\exists mav \in acceptorMsgsOfType(\text{“2av”}) :$
$\wedge\ mav.acc = a$

$\land\ mav.bal\ = b$
$\land\ mav.val\ = m.val$
BY $\langle 5\rangle 2$ DEF $2amsgs$

$\langle 5\rangle 4.$ PICK $Q2 \in Quorum :$
$\forall\, a \in Q2 \quad :$
$\exists\, m2av \in acceptorMsgsOfType(\text{"2av"})' :$
$\land\ m2av.acc = a$
$\land\ m2av.bal\ = b$
$\land\ m2av.val\ = v$
BY $\langle 4\rangle 1$, $MsgsTypeLemmaPrime$, Isa DEF $2amsgs$

$\langle 5\rangle 5.$ PICK $a \in Q \cap Q2 : a \in Acceptor$
BY $QuorumTheorem$

$\langle 5\rangle 6.$ PICK $mav \in acceptorMsgsOfType(\text{"2av"}) :$
$\land\ mav.acc = a$
$\land\ mav.bal\ = b$
$\land\ mav.val\ = m.val$
BY $\langle 5\rangle 3$, $\langle 5\rangle 5$

$\langle 5\rangle 7.$ PICK $m2av \in acceptorMsgsOfType(\text{"2av"})' :$
$\land\ m2av.acc = a$
$\land\ m2av.bal\ = b$
$\land\ m2av.val\ = v$
BY $\langle 5\rangle 4$, $\langle 5\rangle 5$

$\langle 5\rangle 8.$ $mav \in acceptorMsgsOfType(\text{"2av"})'$
BY $\langle 2\rangle 4$ DEF $acceptorMsgsOfType$, $msgsOfType$, $Phase2av$

$\langle 5\rangle 9.$ $m.val = v$
BY $\langle 5\rangle 5$, $\langle 5\rangle 6$, $\langle 5\rangle 7$, $\langle 5\rangle 8$ DEF $2avInv1$, $InvP$, Inv, $acceptorMsgsOfType$, $msgsOfType$

$\langle 5\rangle 10.$ QED
BY $\langle 5\rangle 2$, $\langle 5\rangle 9$ DEF $2amsgs$

$\langle 4\rangle 4.$ QED
BY $\langle 2\rangle 4$, $\langle 3\rangle 1$, $\langle 4\rangle 1$, $\langle 4\rangle 2$ DEF $P!TLANext$, $P!Phase2a$, $Phase2av$, $Ballot$, $P!Ballot$

$\langle 3\rangle 4.$ $\lor\ msgs' = msgs$
$\lor\ (\ \land\ msgs' \neq msgs$
$\land\ \exists\, v \in Value :$
$\land\ [type \mapsto \text{"1c"},\ bal \mapsto b,\ val \mapsto v] \in msgs$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"2a"},\ bal \mapsto b,\ val \mapsto v]\})$
BY $MsgsLemma$, $\langle 2\rangle 4$, $Zenon$ DEF Inv

$\langle 3\rangle 5.$ QED
BY $\langle 3\rangle 2$, $\langle 3\rangle 3$, $\langle 3\rangle 4$

$\langle 2\rangle 5.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$,
$Phase2b(self,\ b)$
PROVE $P!TLANext \lor P!vars' = P!vars$

$\langle 3\rangle 1.$ $PmaxBal[self] \leq b$

$\langle 4\rangle 1.$ $PmaxBal[self] \leq maxBal[self]$
BY $PmaxBalLemma4$ DEF Inv

$\langle 4\rangle 2.$ $maxBal[self] \leq b$

BY $\langle 2\rangle 5$ DEF $Phase2b$

$\langle 4\rangle 3.$ QED

BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $PmaxBalLemma5$ DEF Inv, $TypeOK$, $Ballot$

$\langle 3\rangle 2.$ PICK $v \in Value :$

$\land \exists\, Q \in ByzQuorum :$

$\forall\, a \in Q :$

$\exists\, m \in sentMsgs(\text{“2av”}, b) : \land\, m.val = v$

$\land\, m.acc = a$

$\land\, msgs' = msgs \cup$

$\{[type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$

$\land\, bmsgs' = bmsgs \cup$

$\{[type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]\}$

$\land\, maxVVal' = [maxVVal$ EXCEPT $![self] = v]$

BY $\langle 2\rangle 5$, $MsgsLemma$ DEF Inv

$\langle 3\rangle$ DEFINE $m \triangleq [type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v]$

$m2b \triangleq [type \mapsto \text{“2b”}, acc \mapsto self, bal \mapsto b, val \mapsto v]$

$\langle 3\rangle 3.$ $m \in P!sentMsgs(\text{“2a”}, b)$

$\langle 4\rangle 1.$ PICK $Q \in Quorum :$

$\forall\, a \in Q :$

$\exists\, mm \in sentMsgs(\text{“2av”}, b) : \land\, mm.val = v$

$\land\, mm.acc = a$

BY $\langle 3\rangle 2$, Isa DEF $Quorum$

$\langle 4\rangle 2.$ $m \in 2amsgs$

BY $\langle 4\rangle 1$ DEF $sentMsgs$, $Quorum$, $acceptorMsgsOfType$, $msgsOfType$, $2amsgs$

$\langle 4\rangle 3.$ QED

BY $\langle 4\rangle 2$ DEF $P!sentMsgs$, $msgs$

$\langle 3\rangle 4.$ $PmaxBal' = [PmaxBal$ EXCEPT $![self] = b]$

$\langle 4\rangle 1.$ ASSUME NEW $a \in Acceptor$,

$a \neq self$

PROVE $PmaxBal'[a] = PmaxBal[a]$

BY $\langle 3\rangle 2$, $\langle 4\rangle 1$, $PmaxBalLemma2$, $m2b.acc = self$, $Zenon$

$\langle 4\rangle 2.$ $PmaxBal'[self] = b$

$\langle 5\rangle$ DEFINE $S \triangleq \{mm.bal : mm \in \{ma \in bmsgs :$

$\land\, ma.type \in \{\text{“1b”}, \text{“2b”}\}$

$\land\, ma.acc = self\}\}$

$T \triangleq S \cup \{m2b.bal\}$

$\langle 5\rangle 1.$ $IsFiniteSet(S) \land (S \in$ SUBSET $Ballot)$

BY $PMaxBalLemma3$ DEF Inv

$\langle 5\rangle 2.$ $IsFiniteSet(T) \land (T \in$ SUBSET $Ballot)$

BY $\langle 5\rangle 1$, $FS\_AddElement$

$\langle 5\rangle 3.$ $PmaxBal[self] = MaxBallot(S)$

BY DEF $PmaxBal$, $1bOr2bMsgs$

$\langle 5\rangle 4.$ $PmaxBal'[self] = MaxBallot(T)$

BY $\langle 3\rangle 2$, $Zenon$ DEF $PmaxBal$, $1bOr2bMsgs$

$\langle 5\rangle$ HIDE DEF S

$\langle 5\rangle 5.$CASE $S = \{\}$

$\langle 6\rangle$ $MaxBallot(\{b\}) = b$

BY $FS\_Singleton$, $MaxBallotLemma1$, Isa DEF $Ballot$

$\langle 6\rangle$ QED

BY $\langle 5\rangle 4$, $\langle 5\rangle 5$

$\langle 5\rangle 6.$CASE $S \neq \{\}$

$\langle 6\rangle$ $\forall\, bb \in T : b \geq bb$

BY $\langle 3\rangle 1$, $\langle 5\rangle 1$, $\langle 5\rangle 3$, $MaxBallotProp$, $PmaxBalLemma5$ DEF Inv, $Ballot$

$\langle 6\rangle$ QED

BY $\langle 5\rangle 2$, $\langle 5\rangle 4$, $MaxBallotLemma1$

$\langle 5\rangle 7.$ QED

BY $\langle 5\rangle 5$, $\langle 5\rangle 6$

$\langle 4\rangle 3.$ QED

BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $Zenon$ DEF $PmaxBal$, $1bOr2bMsgs$

$\langle 3\rangle 5.$ $\land\, maxVBal' = [maxVBal$ EXCEPT $![self] = b]$

$\land\, maxVVal' = [maxVVal$ EXCEPT $![self] = m.val]$

BY $\langle 2\rangle 5$, $\langle 3\rangle 2$, $Zenon$ DEF $Phase2b$

$\langle 3\rangle 6.$ QED

BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$, $\langle 3\rangle 4$, $\langle 3\rangle 5$, $Zenon$

DEF $P!TLANext$, $P!Phase2b$, $Ballot$, $P!Ballot$

$\langle 2\rangle 6.$ ASSUME NEW $self \in Acceptor$, NEW $b \in Ballot$,

$LearnsSent(self, b)$

PROVE $P!TLANext \lor P!vars' = P!vars$

$\langle 3\rangle 1.$ PICK $SM \in$ SUBSET $\{m \in msgsOfType(\text{“1c”}) : m.bal = b\}$:

$msgs' = msgs \cup SM$

BY $\langle 2\rangle 6$, $MsgsLemma$ DEF Inv

$\langle 3\rangle$ DEFINE $S \triangleq \{m.val : m \in SM\}$

$\langle 3\rangle 2.$ $S \in$ SUBSET $Value$

BY $BMessageLemma$ DEF Inv, $TypeOK$, $msgsOfType$, $1cMessage$

$\langle 3\rangle 3.$ $msgs' = msgs \cup \{[type \mapsto \text{“1c”}, bal \mapsto b, val \mapsto v] : v \in S\}$

BY $\langle 3\rangle 1$, $BMessageLemma$ DEF Inv, $TypeOK$, $msgsOfType$, $1cMessage$

$\langle 3\rangle 4.$ ASSUME NEW $v \in S$

PROVE $\exists\, Q \in Quorum : P!ShowsSafeAt(Q, b, v)$

$\langle 4\rangle 1.$ PICK $ac \in Acceptor : KnowsSafeAt(ac, b, v)'$

BY $\langle 3\rangle 1$, $MsgsTypeLemmaPrime$ DEF $msgsOfType$, $1cmsgs$

$\langle 4\rangle 2.$ $bmsgs' = bmsgs$

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

$\langle 4\rangle$ DEFINE $Q(BQ) \triangleq BQ \cap Acceptor$

$SS \triangleq \{m \in knowsSent'[ac] : m.bal = b\}$

$SQ(BQ) \triangleq \{1bRestrict(mm) :$

$mm \in \{m \in SS : m.acc \in Q(BQ)\}\}$

$Q1b(BQ) \triangleq \{m \in P!sentMsgs(\text{“1b”}, b) : m.acc \in Q(BQ)\}$

$\langle 4\rangle 3.$ ASSUME NEW $BQ \in ByzQuorum$,

$\forall\, a \in BQ : \exists\, m \in SS : m.acc = a$

PROVE $SQ(BQ) = Q1b(BQ)$

⟨5⟩1. ASSUME NEW $m \in P!sentMsgs(\text{"1b"}, b)$,
$m.acc \in Q(BQ)$
PROVE $m \in SQ(BQ)$
BY ⟨4⟩2, ⟨4⟩3, ⟨5⟩1, *MsgsTypeLemma*
DEF $P!sentMsgs$, *msgs*, *1bmsgs*, *acceptorMsgsOfType*, *msgsOfType*, *1bRestrict*, *InvP*, *Inv*, *knowsSentInv*, *1bInv2*
⟨5⟩2. ASSUME NEW $m \in SS$,
$m.acc \in Q(BQ)$
PROVE $1bRestrict(m) \in Q1b(BQ)$
BY ⟨4⟩2, ⟨5⟩2
DEF *InvP*, *Inv*, *knowsSentInv*, *msgsOfType*, *acceptorMsgsOfType*, *msgs*, *1bmsgs*, $P!sentMsgs$, *1bRestrict*
⟨5⟩3. QED
BY ⟨5⟩1, ⟨5⟩2 DEF *Q1b*, *SQ*
⟨4⟩4.CASE $KnowsSafeAt(ac, b, v)!1!1'$
⟨5⟩1. PICK $BQ \in ByzQuorum : KnowsSafeAt(ac, b, v)!1!1!(BQ)'$
BY ⟨4⟩4
⟨5⟩2. $\forall a \in Q(BQ) : \exists m \in SQ(BQ) : \land m.acc = a$
$\land m.mbal = -1$
BY ⟨5⟩1, *Isa* DEF *1bRestrict*
⟨5⟩3. $\forall m \in SQ(BQ) : m.mbal = -1$
BY ⟨4⟩2, ⟨5⟩2
DEF *InvP*, *Inv*, *knowsSentInv*, *msgsOfType*, *1bRestrict*, *1bInv2*
⟨5⟩4. $SQ(BQ) = Q1b(BQ)$
BY ⟨4⟩3, ⟨5⟩1
⟨5⟩5. $Q(BQ) \in Quorum$
BY DEF *Quorum*
⟨5⟩ HIDE DEF *SS*, *Q*, *SQ*
⟨5⟩ WITNESS $Q(BQ) \in Quorum$
⟨5⟩6. QED
BY ⟨5⟩2, ⟨5⟩3, ⟨5⟩4 DEF $P!ShowsSafeAt$
⟨4⟩5.CASE $KnowsSafeAt(ac, b, v)!1!2'$
⟨5⟩1. PICK $c \in 0 \,..\, (b-1) : KnowsSafeAt(ac, b, v)!1!2!(c)'$
BY ⟨4⟩5
⟨5⟩2. PICK $BQ \in ByzQuorum$:
$\forall a \in BQ : \exists m \in SS : \land m.acc = a$
$\land m.mbal \leq c$
$\land (m.mbal = c) \Rightarrow (m.mval = v)$
BY ⟨5⟩1
⟨5⟩3. $SQ(BQ) = Q1b(BQ)$
BY ⟨5⟩2, ⟨4⟩3
⟨5⟩4. $P!ShowsSafeAt(Q(BQ), b, v)!1!1$
⟨6⟩1. SUFFICES ASSUME NEW $a \in Q(BQ)$
PROVE $\exists m \in Q1b(BQ) : m.acc = a$
OBVIOUS

$\langle 6\rangle 2.$ PICK $m \in SS : m.acc = a$
BY $\langle 5\rangle 2$
$\langle 6\rangle 3.$ $\land\ 1bRestrict(m) \in SQ(BQ)$
$\land\ 1bRestrict(m).acc = a$
BY $\langle 6\rangle 2$ DEF $1bRestrict$
$\langle 6\rangle$.QED
BY $\langle 6\rangle 3$, $\langle 5\rangle 3$
$\langle 5\rangle 5.$ PICK $m1c \in msgs$:
$\land\ m1c = [type \mapsto \text{“1c”},\ bal \mapsto m1c.bal,\ val \mapsto v]$
$\land\ m1c.bal \geq c$
$\land\ m1c.bal \in Ballot$
$\langle 6\rangle 1.$ PICK $WQ \in WeakQuorum$:
$\forall\, a \in WQ : \exists\, m \in SS : \land\ m.acc = a$
$\land\ \exists\, r \in m.m2av :$
$\land\ r.bal \geq c$
$\land\ r.val = v$
BY $\langle 5\rangle 1$
$\langle 6\rangle 2.$ PICK $a \in WQ,\ m \in SS$:
$\land\ a \in Acceptor$
$\land\ m.acc = a$
$\land\ \exists\, r \in m.m2av : \land\ r.bal \geq c$
$\land\ r.val = v$
BY $\langle 6\rangle 1$, BQA
$\langle 6\rangle 4.$ PICK $r \in m.m2av : \land\ r.bal \geq c$
$\land\ r.val = v$
BY $\langle 6\rangle 2$
$\langle 6\rangle 5.$ $\land\ m.bal = b$
$\land\ m \in bmsgs$
$\land\ m.type = \text{“1b”}$
$\land\ r.bal \in Ballot$
BY $\langle 4\rangle 2$, $\langle 6\rangle 2$, $BMessageLemma$
DEF Inv, $InvP$, $TypeOK$, $1bMessage$, $knowsSentInv$, $msgsOfType$
$\langle 6\rangle$.QED
BY $\langle 6\rangle 2$, $\langle 6\rangle 4$, $\langle 6\rangle 5$, $Zenon$ DEF Inv, $1bInv1$
$\langle 5\rangle 6.$ ASSUME NEW $m \in Q1b(BQ)$
PROVE $\land\ m1c.bal \geq m.mbal$
$\land\ (m1c.bal = m.mbal) \Rightarrow (m.mval = v)$
$\langle 6\rangle 1.$ PICK $mm \in SS : \land\ mm.acc = m.acc$
$\land\ mm.mbal \leq c$
$\land\ (mm.mbal = c) \Rightarrow (mm.mval = v)$
BY $\langle 5\rangle 2$
$\langle 6\rangle 2.$ PICK $mm2 \in SS : \land\ mm2.acc = m.acc$
$\land\ m = 1bRestrict(mm2)$
BY $\langle 5\rangle 3$ DEF $1bRestrict$
$\langle 6\rangle 3.$ $\land\ mm = mm2$

$\wedge\ mm2.mbal \in Ballot \cup \{-1\}$

BY $\langle 4\rangle 2$, $\langle 6\rangle 1$, $\langle 6\rangle 2$, $BMessageLemma$

DEF Inv, $InvP$, $TypeOK$, $knowsSentInv$, $1bInv2$, $msgsOfType$, $1bMessage$

$\langle 6\rangle$.QED

$\langle 7\rangle\ \forall\, m1cbal,\ mmbal \in Ballot \cup \{-1\}:$

$mmbal \leq c \wedge m1cbal \geq c \Rightarrow \wedge\ m1cbal \geq mmbal$

$\wedge\ mmbal = m1cbal \Rightarrow mmbal = c$

BY DEF $Ballot$

$\langle 7\rangle$ QED

BY $\langle 5\rangle 5$, $\langle 6\rangle 1$, $\langle 6\rangle 2$, $\langle 6\rangle 3$ DEF $1bRestrict$

$\langle 5\rangle 7$. $P!ShowsSafeAt(Q(BQ),\ b,\ v)!1!2!2!(m1c)$

BY $\langle 5\rangle 5$, $\langle 5\rangle 6$

$\langle 5\rangle$.QED

BY $\langle 5\rangle 4$, $\langle 5\rangle 7$, Isa DEF $P!ShowsSafeAt$, $Quorum$

$\langle 4\rangle 6$. QED

BY $\langle 3\rangle 1$, $\langle 4\rangle 1$, $\langle 4\rangle 4$, $\langle 4\rangle 5$ DEF $KnowsSafeAt$

$\langle 3\rangle 6$. QED

BY $\langle 2\rangle 6$, $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$, $\langle 3\rangle 4$, $Zenon$

DEF $LearnsSent$, $P!Phase1c$, $P!TLANext$, $Ballot$, $P!Ballot$, $PmaxBal$, $1bOr2bMsgs$

$\langle 2\rangle 7$. ASSUME NEW $self \in Ballot$,

$Phase1a(self)$

PROVE $P!TLANext \vee P!vars' = P!vars$

$\langle 3\rangle 1$. $msgs' = msgs \cup \{[type \mapsto \text{“1a”},\ bal \mapsto self]\}$

BY $\langle 2\rangle 7$, $MsgsLemma$ DEF Inv

$\langle 3\rangle 2$. UNCHANGED $\langle PmaxBal,\ maxVBal,\ maxVVal\rangle$

BY $\langle 2\rangle 7$, Isa DEF $Phase1a$, $PmaxBal$, $1bOr2bMsgs$

$\langle 3\rangle$.QED

BY $\langle 3\rangle 1$, $\langle 3\rangle 2$ DEF $P!Phase1a$, $P!TLANext$, $Ballot$, $P!Ballot$

$\langle 2\rangle 8$. ASSUME NEW $self \in Ballot$,

$Phase1c(self)$

PROVE $P!TLANext \vee P!vars' = P!vars$

$\langle 3\rangle 1$. PICK $SS \in$ SUBSET $[type : \{\text{“1c”}\},\ bal : \{self\},\ val : Value]:$

$\wedge\ \forall\, m \in SS : \exists\, a \in Acceptor : KnowsSafeAt(a,\ m.bal,\ m.val)$

$\wedge\ msgs' = msgs \cup SS$

BY $\langle 2\rangle 8$, $MsgsLemma$ DEF Inv

$\langle 3\rangle$ DEFINE $S \triangleq \{m.val : m \in SS\}$

$\langle 3\rangle 2$. $SS = \{[type \mapsto \text{“1c”},\ bal \mapsto self,\ val \mapsto v] : v \in S\}$

OBVIOUS

$\langle 3\rangle 3$. ASSUME NEW $v \in S$

PROVE $\exists\, Q \in Quorum : P!ShowsSafeAt(Q,\ self,\ v)$

$\langle 4\rangle$ DEFINE $m \triangleq [type \mapsto \text{“1c”},\ bal \mapsto self,\ val \mapsto v]$

$\langle 4\rangle 1$. PICK $a \in Acceptor : KnowsSafeAt(a,\ self,\ v)$

BY $\langle 3\rangle 1$

$\langle 4\rangle$ DEFINE $SK \triangleq \{mm \in knowsSent[a] : mm.bal = self\}$

$\langle 4\rangle 2$. ASSUME NEW $BQ \in ByzQuorum$,

$\forall\, ac \in BQ : \exists\, mm \in SK : mm.acc = ac$

PROVE $P!ShowsSafeAt(BQ \cap Acceptor,\ self,\ v)!1!1$

$\langle 5 \rangle$ DEFINE $Q \triangleq BQ \cap Acceptor$

$Q1b \triangleq \{mm \in P!sentMsgs(\text{“1b”},\ self) : mm.acc \in Q\}$

$\langle 5 \rangle$ SUFFICES ASSUME NEW $ac \in BQ \cap Acceptor$

PROVE $\exists\, mm \in Q1b : mm.acc = ac$

OBVIOUS

$\langle 5 \rangle 1.$ PICK $mm \in SK : mm.acc = ac$

BY $\langle 4 \rangle 2$

$\langle 5 \rangle 2.$ $\land\ 1bRestrict(mm) \in P!sentMsgs(\text{“1b”},\ self)$

$\land\ 1bRestrict(mm).acc = ac$

BY $\langle 5 \rangle 1$ DEF $acceptorMsgsOfType,\ msgsOfType,\ 1bmsgs,\ msgs,\ Inv,\ knowsSentInv,$ $1bRestrict,\ P!sentMsgs$

$\langle 5 \rangle$.QED

BY $\langle 5 \rangle 2$

$\langle 4 \rangle 3.$CASE $KnowsSafeAt(a,\ self,\ v)!1!1$

$\langle 5 \rangle 1.$ PICK $BQ \in ByzQuorum :$

$\forall\, ac \in BQ : \exists\, mm \in SK : \ \land\ mm.acc = ac$

$\land\ mm.mbal = -1$

BY $\langle 4 \rangle 3$

$\langle 5 \rangle$ DEFINE $Q \triangleq BQ \cap Acceptor$

$Q1b \triangleq \{mm \in P!sentMsgs(\text{“1b”},\ self) : mm.acc \in Q\}$

$\langle 5 \rangle 2.$ $P!ShowsSafeAt(Q,\ self,\ v)!1!1$

BY $\langle 5 \rangle 1$, $\langle 4 \rangle 2$

$\langle 5 \rangle 3.$ ASSUME NEW $mm \in Q1b$

PROVE $mm.mbal = -1$

BY $\langle 5 \rangle 1$, $MsgsTypeLemma$

DEF $P!sentMsgs,\ 1bmsgs,\ acceptorMsgsOfType,\ msgsOfType,\ 1bRestrict,$ $Inv,\ knowsSentInv,\ 1bInv2,\ 1bRestrict$

$\langle 5 \rangle$.QED

BY $\langle 5 \rangle 2$, $\langle 5 \rangle 3$, $Zenon$ DEF $P!ShowsSafeAt,\ Quorum$

$\langle 4 \rangle 4.$CASE $KnowsSafeAt(a,\ self,\ v)!1!2$

$\langle 5 \rangle 1.$ PICK $c \in 0 \,..\, (self - 1) : KnowsSafeAt(a,\ self,\ v)!1!2!(c)$

BY $\langle 4 \rangle 4$

$\langle 5 \rangle 2.$ PICK $BQ \in ByzQuorum : KnowsSafeAt(a,\ self,\ v)!1!2!(c)!1!(BQ)$

BY $\langle 5 \rangle 1$

$\langle 5 \rangle$ DEFINE $Q \triangleq BQ \cap Acceptor$

$Q1b \triangleq \{mm \in P!sentMsgs(\text{“1b”},\ self) : mm.acc \in Q\}$

$\langle 5 \rangle 3.$ $P!ShowsSafeAt(Q,\ self,\ v)!1!1$

BY $\langle 5 \rangle 2$, $\langle 4 \rangle 2$

$\langle 5 \rangle 4.$ PICK $WQ \in WeakQuorum : KnowsSafeAt(a,\ self,\ v)!1!2!(c)!2!(WQ)$

BY $\langle 5 \rangle 1$

$\langle 5 \rangle 5.$ PICK $ac \in WQ \cap Acceptor :$

$KnowsSafeAt(a,\ self,\ v)!1!2!(c)!2!(WQ)!(ac)$

BY $\langle 5 \rangle 4$, BQA

$\langle 5 \rangle 6.$ PICK $mk \in SK : \land mk.acc = ac$
$\quad \land \exists r \in mk.m2av : \land r.bal \geq c$
$\quad\quad \land r.val = v$

BY $\langle 5 \rangle 5$

$\langle 5 \rangle 7.$ PICK $r \in mk.m2av : \land r.bal \geq c$
$\quad \land r.val = v$

BY $\langle 5 \rangle 6$

$\langle 5 \rangle$ DEFINE $mc \triangleq [type \mapsto \text{"1c"}, bal \mapsto r.bal, val \mapsto v]$

$\langle 5 \rangle 9.$ $mc \in msgs$

BY $\langle 5 \rangle 6$, $\langle 5 \rangle 7$ DEF *Inv*, *1bInv1*, *knowsSentInv*, *msgsOfType*

$\langle 5 \rangle 10.$ ASSUME NEW $mq \in Q1b$
PROVE $\land mc.bal \geq mq.mbal$
$\quad \land (mc.bal = mq.mbal) \Rightarrow (mq.mval = v)$

BY $\langle 5 \rangle 2$, $\langle 5 \rangle 7$, *MsgsTypeLemma*, *BMessageLemma*
DEF *P*!*sentMsgs*, *1bmsgs*, *acceptorMsgsOfType*, *msgsOfType*, *1bRestrict*,
Inv, *TypeOK*, *1bInv2*, *knowsSentInv*, *1bMessage*, *Ballot*

$\langle 5 \rangle 11.$ QED

$\langle 6 \rangle$ $Q \in Quorum$

BY DEF *Quorum*

$\langle 6 \rangle$ WITNESS $Q \in Quorum$

$\langle 6 \rangle$ QED

BY $\langle 5 \rangle 3$, $\langle 5 \rangle 9$, $\langle 5 \rangle 10$ DEF *P*!*ShowsSafeAt*

$\langle 4 \rangle 5.$ QED

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

$\langle 3 \rangle$.QED

BY $\langle 2 \rangle 8$, $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, *Zenon*
DEF *P*!*Phase1c*, *Phase1c*, *PmaxBal*, *1bOr2bMsgs*, *P*!*TLANext*, *Ballot*, *P*!*Ballot*

$\langle 2 \rangle 9.$ ASSUME NEW $self \in FakeAcceptor$,
$FakingAcceptor(self)$
PROVE $P!TLANext \lor P!vars' = P!vars$

$\langle 3 \rangle 1.$ $msgs' = msgs$

BY $\langle 2 \rangle 9$, *MsgsLemma* DEF *Inv*

$\langle 3 \rangle 2.$ $PmaxBal' = PmaxBal$

BY $\langle 2 \rangle 9$, *BQA*, *Zenon* DEF *FakingAcceptor*, *PmaxBal*, *1bOr2bMsgs*

$\langle 3 \rangle$.QED

BY $\langle 2 \rangle 9$, $\langle 3 \rangle 1$, $\langle 3 \rangle 2$ DEF *P*!*vars*, *FakingAcceptor*

$\langle 2 \rangle 10.$ QED

BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$, $\langle 2 \rangle 8$, $\langle 2 \rangle 9$, *NextDef*

$\langle 1 \rangle 3.$ QED

BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, *Invariance*, *PTL* DEF *Spec*, *P*!*Spec*

To see how learning is implemented, we must describe how to determine that a value has been chosen. This is done by the following definition of *chosen* to be the set of chosen values.

$$chosen \triangleq \{v \in Value : \exists\, BQ \in ByzQuorum,\ b \in Ballot : \forall\, a \in BQ : \exists\, m \in msgs : \land\ m.type = \text{“2b”} \land\ m.acc = a \land\ m.bal = b \land\ m.val = v\}$$

The correctness of our definition of *chosen* is expressed by the following theorem, which asserts that if a value is in *chosen* , then it is also in the set *chosen* of the emulated execution of the *PCon* algorithm.

The state function *chosen* does not necessarily equal the corresponding state function of the *PCon* algorithm. It requires every (real or fake) acceptor in a *ByzQuorum* to vote for (send $2b$ messages) for a value v in the same ballot for v to be in *chosen* for the *BPCon* algorithm, but it requires only that every (real) acceptor in a *Quorum* vote for v in the same ballot for v to be in the set *chosen* of the emulated execution of algorithm *PCon*.

Liveness for *BPCon* requires that, under suitable assumptions, some value is eventually in *chosen* . Since we can't assume that a fake acceptor does anything useful, liveness requires the assumption that there is a *ByzQuorum* composed entirely of real acceptors (the first conjunct of assumption *BQLA*).

THEOREM $chosen \subseteq P!chosen$

BY *Isa* DEF $chosen,\ P!chosen,\ Quorum,\ Ballot,\ P!Ballot$

---

* Modification History
* Last modified *Fri Jul* 24 17:51:34 *CEST* 2020 by *merz*
* Last modified *Wed Apr* 15 15:16:26 *CEST* 2020 by *doligez*
* Last modified *Mon Aug* 18 14:57:27 *CEST* 2014 by *tomer*
* Last modified *Mon Mar* 04 17:24:05 *CET* 2013 by *doligez*
* Last modified *Wed Nov* 30 15:47:26 *PST* 2011 by *lamport*
* Last modified *Wed Dec* 01 11:35:29 *PST* 2010 by *lamport*