Proving the Correctness of Disk Paxos in Isabelle/HOL

Mauro Jaskelioff  Stephan Merz

August 28, 2014

Abstract

Disk Paxos [GL00] is an algorithm for building arbitrary fault-tolerant distributed systems. The specification of Disk Paxos has been proved correct informally and tested using the TLC model checker, but up to now, it has never been fully formally verified. In this work we have formally verified its correctness using the Isabelle theorem prover and the HOL logic system [NPW02], showing that Isabelle is a practical tool for verifying properties of TLA⁺ specifications.

Contents

1 Introduction 2

2 The Disk Paxos Algorithm 3
   2.1 Informal description of the algorithm 4
   2.2 Disk Paxos and its TLA⁺ Specification 4

3 Translating from TLA⁺ to Isabelle/HOL 6
   3.1 Typed vs. Untyped 6
   3.2 Primed Variables 8
   3.3 Restructuring the specification 8

4 Structure of the Correctness Proof 9
   4.1 Going from Informal Proofs to Formal Proofs 10

5 Conclusion 11

A TLA⁺ correctness specification 12

B Disk Paxos Algorithm Specification 13
1 Introduction

Algorithms for fault-tolerant distributed systems were first introduced to implement critical systems. Nevertheless, what good is such an algorithm if it has a design error? We need some kind of guarantee that the algorithm does not have a faulty design. Formal verification of its specification is one such guarantee.

Disk Paxos [GL00] is an algorithm for building arbitrary fault-tolerant distributed systems, that, due to its complexity, is difficult to reason about. It has been proved correct informally, and tested using the TLC model checker [Lam02]. The informal proof is rigorous but, as it is always the case with large informal proofs, it is easy to overlook details. Thus, one of the motivations of this work is to see if such a rigorous proof can be formalized in a contemporary theorem prover.

In [Pac01] part of the correctness proof (invariance of \(Hinv1\) and \(Hinv3\)) was verified using the theorem prover ACL2 [KMM00]. An implicit assumption of this formalization is that all sets are finite, thus overlooking the fact that there is a missing conjunct in the Disk Paxos invariant (see Section 4).

We set the goal of formally verifying Disk Paxos correctness using the theorem prover Isabelle/HOL [NPW02]. In this way, we could gain more confidence in the correctness of Disk Paxos design and, at the same time, learn to what extent can Isabelle be a useful tool for proving the correctness of distributed systems using a real world example.
In Section 2 we give a brief description of the algorithm and its specification. In Section 3 we describe the translation from TLA+ to Isabelle/HOL and the problems that this translation originated. In Section 4 we discuss how our formal proofs relate to the informal ones in [GL00], and in Section 5 we conclude. The entire specification and all formal proofs can be found in the Appendix.

2 The Disk Paxos Algorithm

Disk Paxos is a variant of the classic Paxos algorithm [Lam98] for the implementation of arbitrary fault-tolerant systems with a network of processors and disks. It maintains consistency in the event of any number of non-Byzantine failures. This means that a processor may fail completely or pause for arbitrary long periods and then restart, remembering only that it has failed. A disk may become inaccessible to some or all processors, but it may not be corrupted. We say that a system is stable if all processes are either non-faulty or have failed completely (i.e. there are no new failed processes). Disk Paxos guarantees progress if the system is stable and there is at least one non-faulty processor that can read and write a majority of the disks. Consequently, the fundamental difference between Classic Paxos and Disk Paxos is that the former achieves redundancy by replicating processes while the latter replicates disks. Since disks are usually cheaper than processors, it is possible to obtain more redundancy at a lower cost.

Disk Paxos uses the state machine approach to solve the problem of implementing an arbitrary distributed system. The state machine approach [Sch90] is a general method that reduces this problem to solving a consensus problem. The distributed system is designed as a deterministic state machine that executes a sequence of commands, and a consensus algorithm ensures that, for each $n$, all processors agree on the $n^{th}$ command. Hence, each processor $p$ starts with an input value (a command), and it may output a value (the command to be executed). The problem is solved if all processors eventually output the same value and this value was a value of $\text{input}[p]$ for some $p$ (under certain assumptions, in our case that there exists at least one non-faulty processor that can write and read a majority of disks).

Progress of Disk Paxos relies on progress of a leader-election algorithm. It is easy to make a leader-election algorithm if the system is stable, but very hard to devise one that works correctly even if the system is unstable. We are requiring stability to ensure progress, but actually only a weaker requirement is needed: progress of the underlying leader-election algorithm. Disk Paxos ensures that all outputs (if any) will be the same even if the leader-election algorithm fails.
2.1 Informal description of the algorithm.

The consensus algorithm of Disk Paxos is called Disk Synod. In it, each processor has an assigned block on each disk. Also it has a local memory that contains its current block (called the dblock), and other state variables (see figure 1). When a process \( p \) starts it contains an input value \( \text{input}[p] \) that will not be modified, except possibly when recovering from a failure.

Disk Synod is structured in two phases, plus one more phase for recovering from failures. In each phase, a processor writes its own block and reads each other processor’s block, on a majority of the disks. The idea is to execute ballots to determine:

**Phase 1:** whether a processor \( p \) can choose its own input value \( \text{input}[p] \) or must choose some other value. When this phase finishes a value \( v \) is chosen.

**Phase 2:** whether it can commit \( v \). When this phase is complete the process has committed value \( v \) and can output it (using variable \( \text{outpt} \)).

In either phase, a processor aborts its ballot if it learns that another processor has begun a higher-numbered ballot. The third phase (Phase 0) is for starting the algorithm or recovering from a failure.

In each block, processors maintain three values:

- **mbal** The current ballot number.
- **bal** The largest ballot number for which the processor entered phase 2.
- **inp** The value the processor tried to commit in ballot number \( \text{bal} \).

For a complete description of the algorithm, see [GL00].

2.2 Disk Paxos and its TLA\(^+\) Specification

The specification of Disk Paxos is written in the TLA\(^+\) specification language [Lam02]. As it is usual with TLA\(^+\), the specification is organized into modules.

The specification of consensus is given in module Synod, which can be found in appendix A. In it there are only two variables: \( \text{input} \) and \( \text{output} \). To formalize the property stating that all processors should choose the same value and that this value should have been an input of a processor, we need variables that represent all past inputs and the value chosen as result. Consequently, an Inner submodule is introduced, which adds two variables: \( \text{allInput} \) and \( \text{chosen} \). Our Synod module will be obtained by existentially quantifying these variables of the Inner module.

The specification of the algorithm is given in the HDiskSynod module. Hence, what we are going to prove is that the (translation to Isabelle/HOL
of the) Inner module is implied by the (translation to Isabelle/HOL of the) algorithm module HDiskSynod.

More concretely we have that the specification of the algorithm is:

\[
\text{HDiskSynodSpec} \equiv \text{HInit} \land \Box [\text{HNext}]_{\text{vars}, \text{chosen}, \text{allInput}}
\]

where HInit describes the initial state of the algorithm and HNext is the action that models all of its state transitions. The variable vars is the tuple of all variables used in the algorithm.

Analogously, we have the specification of the Inner module:

\[
\text{ISpec} \equiv \text{IInit} \land \Box [\text{INext}]_{\text{input}, \text{output}, \text{chosen}, \text{allInput}}
\]

We define ivars = (input, output, chosen, allInput). In order to prove that HDiskSynodSpec implies ISpec, we follow the structure of the proof given by Gafni and Lamport. We must prove two theorems:

**THEOREM R1**  \(\text{HInit} \Rightarrow \text{IInit}\)

**THEOREM R2**  \(\text{HInit} \land \Box [\text{HNext}]_{\text{vars}, \text{chosen}, \text{allInput}} \Rightarrow \Box [\text{INext}]_{\text{ivars}}\)

The proof of R1 is trivial. For R2, we use TLA proof rules [Lam02] that show that to prove R2, it suffices to find a state predicate HI\text{nv} for which we can prove:

**THEOREM R2a**  \(\text{HInit} \land \Box [\text{HNext}]_{\text{vars}, \text{chosen}, \text{allInput}} \Rightarrow \Box \text{HI\text{nv}}\)

**THEOREM R2b**  \(\text{HI\text{nv}} \land \text{HI\text{nv}'} \land \text{HNext} \Rightarrow \text{INext} \lor (\text{UNCHANGED ivars})\)

A predicate satisfying HI\text{nv} is said to be an invariant of HDiskSynodSpec. To prove R2a, we make HI\text{nv} strong enough to satisfy:
\[
\exists d \in D \, : \, \text{disk}[d][q]. \text{bal} = \text{bk}
\]

\[
\exists d \in D. \, \text{bal}(\text{disk } s \, d \, q) = \text{bk}
\]

\[
\text{choose } x. P x
\]

\[
\text{phase}' = \text{phase} \setminus \{p\} = 1
\]

\[
\text{phase}' = (\text{phase } s)(p := 1)
\]

\[
\text{UN } p. \, \text{blocksOf } s \, p
\]

\[
\text{UNCHANGED } v
\]

\[
v' = v
\]

Table 1: Examples of TLA\(^+\) formulas and their counterparts in Isabelle/HOL.

**THEOREM I1**  \( HInit \Rightarrow HInv \)

**THEOREM I2**  \( HInv \land H\text{Next} \Rightarrow HInv' \)

Again, we have TLA proof rules that say that \( I1 \) and \( I2 \) imply \( R2a \). In summary, \( R2b, I1, \) and \( I2 \) together imply \( HDiskSynodSpec \Rightarrow ISpec \).

Finding a predicate \( HInv \) that is strong enough can be rather difficult. Fortunately, Gafni and Lamport give this predicate. In their paper, they present \( HInv \) as a conjunction of 6 predicates \( HInv_1, \ldots, HInv_6 \), where \( HInv_1 \) is a simple “type invariant” and the higher-numbered predicates add more and more information. The proof is structured such that the preservation of \( HInv_i \) by the algorithm’s next-state relation relies on all \( HInv_j \) (for \( j \leq i \)) being true in the state before the transition. In our proofs we are going to use exactly the same tactic.

Before starting our proofs we have to translate all the specification and theorems above into the formal language of Isabelle/HOL.

### 3 Translating from TLA\(^+\) to Isabelle/HOL

The translation from TLA\(^+\) to Isabelle/HOL is pretty straightforward as Isabelle/HOL has equivalent counterparts for most of the constructs in TLA\(^+\) (some representative examples are shown in table 1). Nevertheless, there are some semantic discrepancies. In the following, we discuss these differences, some of the options that one has when dealing with them, and the reasons for our choices\(^1\).

#### 3.1 Typed vs. Untyped

TLA\(^+\) is an untyped formalism. However, TLA\(^+\) specifications usually have some type information, usually in the form of set membership or set inclusion. When translating these specifications to Isabelle/HOL, which is a typed formalism, one has to invent types that represent these sets of values.

---

\(^1\)There is no point in using the existing TLA encoding in Isabelle. Since the encoding is also based on HOL and we only prove safety, we would have gained nothing.
TLA⁺:

**CONSTANT** Inputs

\[ \text{NotAnInput} \triangleq \text{CHOOSE } c : c \notin \text{Inputs} \]

\[ \text{DiskBlock} \triangleq [\text{mbal} : (\text{UNION Ballot}(p) : p \in \text{Proc}) \cup \{0\}, \]
\[ \text{bal} : (\text{UNION Ballot}(p) : p \in \text{Proc}) \cup \{0\}, \]
\[ \text{inp} : \text{Inputs} \cup \{\text{NotAnInput}\}] \]

**Isabelle/HOL:**

**typedef** InputsOrNi

**consts**

\[ \text{Inputs} :: \text{InputsOrNi set} \]
\[ \text{NotAnInput} :: \text{InputsOrNi} \]

**axioms**

\[ \text{NotAnInput} : \text{NotAnInput} \notin \text{Inputs} \]
\[ \text{InputsOrNi}: (\text{UNIV} :: \text{InputsOrNi set}) = \text{Inputs} \cup \{\text{NotAnInput}\} \]

**record**

\[ \text{DiskBlock} = \]
\[ \text{mbal} :: \text{nat} \]
\[ \text{bal} :: \text{nat} \]
\[ \text{inp} :: \text{InputsOrNi} \]

Figure 2: Untyped TLA⁺ vs. Typed Isabelle/HOL

This process is not automatic and requires some thought to find the right abstractions. Furthermore, simple types may not be expressive enough to represent exactly the set in the specification. In some cases, these sets could be modelled by algebraic datatypes, but this would make the specification more complex and less directly related to the original one. In this work, we have chosen to stick to simple types and add additional axioms to account for their lack of expressiveness.

For example, see figure 2. The type \text{InputsOrNi} models the members of the set \text{Inputs}, and the element \text{NotAnInput}. We record the fact that \text{NotAnInput} is not in \text{Inputs}, with axiom \text{NotAnInput}. Now, looking at the type of the \text{inp} field of the \text{DiskBlock} record in the TLA⁺ specification, we see that its type should be \text{InputsOrNi}. However, this is not the same type as \text{Inputs} \cup \{\text{NotAnInput}\}, as nothing prevents the \text{InputsOrNi} type from having more values. Consequently, we add the axiom \text{InputsOrNi} to establish that the only values of this type are the ones in \text{Inputs} and \text{NotAnInput}.

This example shows the kind of difficulties that can arise when trans-
TLA⁺:

\[
\text{Phase1or2Write}(p, d) \triangleq \\
\land \text{phase}[p] \in \{1, 2\} \\
\land \text{disk}' = [\text{disk EXCEPT } ![d][p] = \text{dblock}[p]] \\
\land \text{disksWritten'} = [\text{disksWritten EXCEPT } ![p] = @ \cup \{d\}] \\
\land \text{UNCHANGED } (\text{input}, \text{output}, \text{phase}, \text{dblock}, \text{blocksRead})
\]

Isabelle/HOL:

\[
\text{Phase1or2Write} :: \text{state } \Rightarrow \text{state } \Rightarrow \text{Proc } \Rightarrow \text{Disk } \Rightarrow \text{bool} \\
\text{Phase1or2Write } s \ s' \ p \ d \equiv \\
\land \text{phase } s \ p \in \{1, 2\} \\
\land \text{disk } s' = (\text{disk } s) (d := (\text{disk } s \ d) (p := \text{dblock } s \ p)) \\
\land \text{disksWritten } s' = (\text{disksWritten } s) (p := (\text{disksWritten } s \ p) \cup \{d\}) \\
\land \text{inpt } s' = \text{inpt } s \land \text{outpt } s' = \text{outpt } s \\
\land \text{phase } s' = \text{phase } s \land \text{dblock } s' = \text{dblock } s \\
\land \text{blocksRead } s' = \text{blocksRead } s
\]

Figure 3: Translation of an action

3.2 Primed Variables

In TLA⁺, to denote the value of a variable in the state resulting from an action, there exists the concept of primed variables. In Isabelle/HOL, there is no built-in notion of state, so we define the state as a record of the variables involved. Instead of defining a “priming” operator, we just define actions as predicates that take any two states, intended to be the previous state and the next state. In this way, \( P \ s \ s' \) will be true iff executing an action \( P \) in the \( s \) state could result in the \( s' \) state. In figure 3 we can see how the action \( \text{Phase1or2Write} \) is expressed in TLA⁺ and in Isabelle/HOL.

3.3 Restructuring the specification

There are some minor changes that can be made to specifications to make the proofs in Isabelle easier.

One such change is the elimination of \textit{let} constructs by making them global definitions. This has two benefits, as it makes it possible to:

- Formulate lemmas that use these definitions.
- Selectively unfold these definitions in proofs, instead of adding \textit{Let-def} to Isabelle’s simplifier, which unfolds all “let” constructs.
Another change that makes proofs easier is to break down actions into simpler actions (e.g. giving separate definitions for subactions corresponding to disjuncts). By making actions smaller, Isabelle has to deal with smaller formulas when we feed the prover with such an action.

For example, Phase1or2Read is mainly a big if-then-else. We break it down into two simpler actions:

\[ \text{Phase1or2Read} \triangleq \text{Phase1or2ReadThen} \lor \text{Phase1or2ReadElse} \]

In Phase1or2ReadThen the condition of the if-then-else is present as a state formula (i.e. it is an enabling condition) while in Phase1or2ReadElse we add the negation of this condition.

Another example is HInv2, which we break down into:

\[ \text{HInv2} \triangleq \text{Inv2a} \land \text{Inv2b} \land \text{Inv2c} \]

Not only we break it down into three conjuncts but, since these conjuncts are quantifications over some predicate, we give a separate definition for this predicate. For example for Inv2a, and after translating to Isabelle/HOL, instead of writing:

\[ \text{Inv2a } s \equiv \forall p. \forall bk \in \text{blocksOf } s \ p. \ldots \]

we write:

\[ \text{Inv2a-innermost} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlock} \Rightarrow \text{bool} \]

\[ \text{Inv2a-inner} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

Now we can express that we want to obtain the fact

\[ \text{Inv2a-innermost } s \ q \ (\text{dblock } s \ q) \]

explicitly stating that we are interested in predicate Inv2a, but only for some process \( q \) and block \( (\text{dblock } s \ q) \).

\section*{4 Structure of the Correctness Proof}

In [GL00], a specification of correctness and a specification of the algorithm are given. Then, it is proved that the specification of the algorithm implies the specification of correctness. We will do the same for the translated specifications, maintaining the names of theorems and lemmas. It should be noted that only safety properties are given and proved.
4.1 Going from Informal Proofs to Formal Proofs

There are informal proofs for invariants $HInv3-HInv6$ and for theorem $R2b$ in [GL00]. These informal proofs are written in the structured-proof style that Lamport advocates [Lam95], and are rigorous and quite detailed. We based our formal proofs on these informal proofs, but in many cases a higher level of detail was needed. In some cases, the steps where too big for Isabelle to solve them automatically, and intermediate steps had to be proved first; in other cases, some of the facts relevant to the proofs were omitted in the informal proofs.

As an example of these omissions, the invariant should state that the set $allRdBlks$ is finite. This is needed to choose a block with a maximum ballot number in action $EndPhase1$. Interestingly, this omission cannot be detected with finite-state model checking. As another example, it was omitted that it is necessary to assume that $HInv4$ and $HInv5$ hold in the previous state to prove lemma $I2f$.

Although our proofs were based on the informal ones, the high-level structure was often different. When proving that a predicate $I$ was an invariant of $Next$, we preferred proving the invariance of $I$ for each action, rather than a big theorem proving the invariance of $I$ for the $Next$ action. As a consequence, a proof for some actions was often similar to the proof of some other action. For example, the proof of the invariance of $HInv3$ for the $EndPhase0$ and $Fail$ actions is almost the same. This means we could have made only one proof for the two actions, shortening the length of the complete proof. Nevertheless, proving each action separately could be tackled more easily by Isabelle, as it had fewer facts to deal with, and proving a new lemma was often a simple matter of copy, paste and renaming of definitions. Once the invariance of a given predicate has been proved for each simple action, proving it for the $Next$ action is easy since the $Next$ action is a disjunction of all actions.

The informal proofs start working with $Next$, and then do a case split in which each case implies some action. The structure of the formal proof was copied from the informal one, in the parts where the latter focused on a particular action. This structure could be easily maintained since we used Isabelle’s Isar proof language [Wen02, Nip03], a language for writing human-readable structured proofs.

Lamport’s use of a hierarchical scheme for naming subformulas of a formula would have been very useful, as we have to repeatedly extract subfacts from facts. These proofs were always solved automatically by the auto Isabelle tactic, but made the proofs longer and harder to understand. Automatic naming of subformulas of Isabelle definitions would be a very practical feature.
5 Conclusion

We formally verified the correctness of the Disk Paxos specification in Isabelle/HOL. We found some omissions in the informal proofs, including one that could not have been detected with finite-state model checking, but no outright errors. This formal proof gives us a greater confidence in the correctness of the algorithm.

This work was done in little more than two months, including learning Isabelle from scratch. Consequently, this work can be taken as evidence that formal verification of fault-tolerant distributed algorithms may be feasible for software verification in an industrial context.

Isabelle proved to be a very helpful tool for this kind of verification, although it would have been useful to have Lamport’s naming of subfacts to make proofs shorter and easier to write.

References


A TLA$^+$ correctness specification

---

**MODULE Synod**

EXTENDS Naturals  
CONSTANT $N$, Inputs  
ASSUME $(N \in \text{Nat}) \land (N > 0)$  
$Proc \triangleq 1..N$  
$NotAnInput \triangleq \text{choose } c : c \not\in \text{Inputs}$  
VARIABLES inputs, output

---

**MODULE Inner**

VARIABLES allInput, chosen

---

$IInit \triangleq \land \text{input} \in [Proc \rightarrow \text{Inputs}]$  
$\land \text{output} = \{p \in \text{Proc} \leftrightarrow \text{NotAnInput}\}$  
$\land \text{chosen} = \text{NotAnInput}$  
$\land \text{allInput} = \text{input}[p] : p \in \text{Proc}$

$IChoose(p) \triangleq \land \text{output}[p] = \text{NotAnInput}$  
$\land \text{IF chosen} = \text{NotAnInput}$  
$\text{THEN } ip \in \text{allInput} : \land \text{chosen}' = ip$  
$\land \text{output}' = [\text{output} \text{ except } !p = ip]$  
$\text{ELSE } \land \text{output}' = [\text{output} \text{ except } !p = \text{chosen}]$  
$\land \text{UNCHANGED chosen}$  
$\land \text{UNCHANGED } \langle \text{input}, \text{allInput} \rangle$

$IFail(p) \triangleq \land \text{output}' = [\text{output} \text{ except } !p = \text{NotAnInput}]$  
$\land \exists ip \in \text{Inputs} : \land \text{input}' = [\text{input} \text{ except } !p = ip]$  
$\land \text{allInput}' = \text{allInput} \cup \{ip\}$

$INext \triangleq \exists p \in \text{Proc} : IChoose(p) \lor IFail(p)$  
$ISpec \triangleq IInit \land \square [INext](input, output, chosen, allInput)$

$IS(\text{chosen}, \text{allInput}) \triangleq \text{instance Inner}$  
$\text{SynodSpec} \triangleq \exists \text{chosen, allInput} : IS(\text{chosen}, \text{allInput}) \land ISpec$
B  Disk Paxos Algorithm Specification

theory DiskPaxos-Model imports Main begin

This is the specification of the Disk Synod algorithm.

typedec InputsOrNi
typedec Disk
typedec Proc

axiomatization

Inputs :: InputsOrNi set and
NotAnInput :: InputsOrNi and
Ballot :: Proc ⇒ nat set and
IsMajority :: Disk set ⇒ bool

where

NotAnInput: NotAnInput ∉ Inputs and
InputsOrNi: (UNIV :: InputsOrNi set) = Inputs ∪ {NotAnInput} and
Ballot-nzero: ∀ p. 0 ∉ Ballot p and
Ballot-disj: ∀ p q. p ≠ q → (Ballot p) ∩ (Ballot q) = {} and
Disk-isMajority: IsMajority(UNIV) and
majorities-intersect:
∀ S T. IsMajority(S) ∧ IsMajority(T) → S ∩ T ≠ {}

lemma ballots-not-zero [simp]:
b ∈ Ballot p =⇒ 0 < b

proof (rule contr)
assume b: b ∈ Ballot p and contr: ¬ (0 < b)
from Ballot-nzero have 0 ∉ Ballot p ..
with b contr show False
  by auto
qed

lemma majority-nonempty [simp]: IsMajority(S) =⇒ S ≠ {}
proof(auto)
from majorities-intersect
have IsMajority(\{}\) ∧ IsMajority(\{}) → {} ∩ {} ≠ {} by auto
thus IsMajority {} =⇒ False by auto
qed

definition AllBallots :: nat set
where AllBallots = (UN p. Ballot p)

record
DiskBlock =

13
mbal :: nat
bal :: nat
inp :: InputsOrNi

**definition** InitDB :: DiskBlock
**where** InitDB = ( mbal = 0, bal = 0, inp = NotAnInput )

**record**
BlockProc =
  block :: DiskBlock
  proc :: Proc

**record**
state =
  inpt :: Proc ⇒ InputsOrNi
  outpt :: Proc ⇒ InputsOrNi
  disk :: Disk ⇒ Proc ⇒ DiskBlock
  dblock :: Proc ⇒ DiskBlock
  phase :: Proc ⇒ nat
  disksWritten :: Proc ⇒ Disk set
  blocksRead :: Proc ⇒ Disk ⇒ BlockProc set

allInput :: InputsOrNi set
chosen :: InputsOrNi

**definition** hasRead :: state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool
**where** hasRead s p d q = (∃ br ∈ blocksRead s p d. proc br = q)

**definition** allRdBlks :: state ⇒ Proc ⇒ BlockProc set
**where** allRdBlks s p = ( UN d. blocksRead s p d )

**definition** allBlocksRead :: state ⇒ Proc ⇒ DiskBlock set
**where** allBlocksRead s p = block ' (allRdBlks s p)

**definition** Init :: state ⇒ bool
**where**
  Init s =
    ( range (inpt s) ⊆ Inputs
    & outpt s = (λp. NotAnInput)
    & disk s = (λd p. InitDB)
    & phase s = (λp. 0)
    & dblock s = (λp. InitDB)
    & disksWritten s = (λp. {}))
    & blocksRead s = (λp d. {}))

**definition** InitializePhase :: state ⇒ state ⇒ Proc ⇒ bool
**where**
InitializePhase s s' p =
\[(\text{disksWritten } s') = (\text{disksWritten } s)(p := \{\}) \]
\& \(\text{blocksRead } s' = (\text{blocksRead } s)(p := (\lambda d. \{\}))\)

\textbf{definition} \(\text{StartBallot} :: \text{state} \to \text{state} \to \text{Proc} \to \text{bool}\)
\textbf{where}
\(\text{StartBallot } s s' p =\)
\begin{align*}
& (\text{phase } s p \in \{1, 2\}) \\
& \& (\exists b \in \text{Ballot } p,
\quad \text{mbal}(\text{dblock } s p) < b)
\quad \& \quad \text{dblock } s' = (\text{dblock } s)(p := (\text{dblock } s p)(\text{mbal} := b |))
\end{align*}
\& \(\text{InitializePhase } s s' p \& \inpt s' = \inpt s \& \text{outpt } s' = \text{outpt } s \& \text{disk } s' = \text{disk } s)\)

\textbf{definition} \(\text{Phase1or2Write} :: \text{state} \to \text{state} \to \text{Proc} \to \text{Disk} \to \text{bool}\)
\textbf{where}
\(\text{Phase1or2Write } s s' p d =\)
\begin{align*}
& (\text{phase } s p \in \{1, 2\}) \\
& \& \text{disk } s' = (\text{disk } s)(d := (\text{disk } s d)(p := (\text{disk } s p))) \\
& \& \text{disksWritten } s' = (\text{disksWritten } s)(p := (\text{disksWritten } s)(p := (\text{disksWritten } s \cup \{d\})))
\end{align*}
\& \(\text{inpt } s' = \inpt s \& \text{outpt } s' = \text{outpt } s \& \text{disk } s' = \text{disk } s \\
\& \text{blocksRead } s' = \text{blocksRead } s)\)

\textbf{definition} \(\text{Phase1or2ReadThen} :: \text{state} \to \text{state} \to \text{Proc} \to \text{Disk} \to \text{Proc} \to \text{bool}\)
\textbf{where}
\(\text{Phase1or2ReadThen } s s' p d q =\)
\begin{align*}
& (d \in \text{disksWritten } s p) \\
& \& \text{mbal}(\text{disk } s d q) < \text{mbal}(\text{dblock } s p) \\
& \& \text{blocksRead } s' = (\text{blocksRead } s)(p := (\text{blocksRead } s)(d := (\text{blocksRead } s)(p := (\text{blocksRead } s)(p := (\text{blocksRead } s)(d := (\text{blocksRead } s)(p := (\text{blocksRead } s)(p := \{\| \text{block} = (\text{disk } s d q, \\
\quad \text{proc} := q |\})\}))))) \\
& \& \text{inpt } s' = \inpt s \& \text{outpt } s' = \text{outpt } s \\
& \& \text{disk } s' = \text{disk } s \& \text{phase } s' = \text{phase } s \\
& \& \text{dblock } s' = \text{dblock } s \& \text{disksWritten } s' = \text{disksWritten } s)\)

\textbf{definition} \(\text{Phase1or2ReadElse} :: \text{state} \to \text{state} \to \text{Proc} \to \text{Disk} \to \text{Proc} \to \text{bool}\)
\textbf{where}
\(\text{Phase1or2ReadElse } s s' p d q =\)
\begin{align*}
& (d \in \text{disksWritten } s p) \\
& \& \text{StartBallot } s s' p
\end{align*}

\textbf{definition} \(\text{Phase1or2Read} :: \text{state} \to \text{state} \to \text{Proc} \to \text{Disk} \to \text{Proc} \to \text{bool}\)
\textbf{where}
\(\text{Phase1or2Read } s s' p d q =\)
\(\text{Phase1or2ReadThen } s s' p d q \lor \text{Phase1or2ReadElse } s s' p d q\)

\textbf{definition} \(\text{blocksSeen} :: \text{state} \to \text{Proc} \to \text{DiskBlock set}\)

15
where \( \text{blocksSeen} \ s \ p = \text{allBlocksRead} \ s \ p \cup \{\text{dblock} \ s \ p\} \)

definition \( \text{nonInitBlks} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlk set} \)

where \( \text{nonInitBlks} \ s \ p = \{bs \ . \ bs \in \text{blocksSeen} \ s \ p \land \text{inp} \ bs \in \text{Inputs}\} \)

definition \( \text{maxBlk} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlk} \)

where
\[
\text{maxBlk} \ s \ p = (\text{SOME} \ b. \ b \in \text{nonInitBlks} \ s \ p \land (\forall c \in \text{nonInitBlks} \ s \ p. \ \text{bal} \ c \leq \text{bal} \ b))
\]

definition \( \text{EndPhase1} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)

where
\[
\text{EndPhase1} \ s \ s' \ p = \\
\{ \text{IsMajority} \ \{d . d \in \text{disksWritten} \ s \ p \} \land (\forall q \in \text{UNIV} - \{p\}. \ \text{hasRead} \ s \ p \ d \ q) \}
\land \text{phase} \ s \ p = 1
\land \text{dblock} \ s' = (\text{dblock} \ s) \ (p := \text{dblock} \ s \ p)
\land \text{inp} :=
\{ \text{if} \ \text{nonInitBlks} \ s \ p = \{\}
\text{then} \ \text{inpt} \ s \ p
\text{else} \ \text{inp} \ (\text{maxBlk} \ s \ p))
\}
\land \text{outpt} \ s' = \text{outpt} \ s
\land \text{phase} \ s' = (\text{phase} \ s) \ (p := \text{phase} \ s \ p + 1)
\land \text{InitializePhase} \ s \ s' \ p
\land \text{inpt} \ s' = \text{inpt} \ s \land \text{disk} \ s' = \text{disk} \ s
\]

definition \( \text{EndPhase2} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)

where
\[
\text{EndPhase2} \ s \ s' \ p = \\
\{ \text{IsMajority} \ \{d . d \in \text{disksWritten} \ s \ p \} \land (\forall q \in \text{UNIV} - \{p\}. \ \text{hasRead} \ s \ p \ d \ q) \}
\land \text{phase} \ s \ p = 2
\land \text{outpt} \ s' = (\text{outpt} \ s) \ (p := \text{inp} \ (\text{dblock} \ s \ p))
\land \text{dblock} \ s' = \text{dblock} \ s
\land \text{phase} \ s' = (\text{phase} \ s) \ (p := \text{phase} \ s \ p + 1)
\land \text{InitializePhase} \ s \ s' \ p
\land \text{inpt} \ s' = \text{inpt} \ s \land \text{disk} \ s' = \text{disk} \ s
\]

definition \( \text{EndPhase1or2} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)

where
\[
\text{EndPhase1or2} \ s \ s' \ p = (\text{EndPhase1} \ s \ s' \ p \lor \text{EndPhase2} \ s \ s' \ p)
\]

definition \( \text{Fail} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)

where
\[
\text{Fail} \ s \ s' \ p = \\
(\exists \ ip \in \text{Inputs}. \ \text{inpt} \ s' = (\text{inpt} \ s) \ (p := ip)
\land \text{phase} \ s' = (\text{phase} \ s) \ (p := 0)
\land \text{dblock} \ s' = (\text{dblock} \ s) \ (p := \text{InitDB})
\]
\begin{align*}
&\land \text{outpt } s' = (\text{outpt } s) \ (p := \text{NotAnInput}) \\
&\land \text{InitializePhase } s \ s' \ p \\
&\land \text{disk } s' = \text{disk } s) \\
\end{align*}

**definition** Phase0Read :: state $\Rightarrow$ state $\Rightarrow$ Proc $\Rightarrow$ Disk $\Rightarrow$ bool

**where**

\begin{align*}
\text{Phase0Read } s \ s' \ p \ d = \\
&\ (\text{phase } s \ p = 0) \\
&\land \text{blocksRead } s' = (\text{blocksRead } s) \ (p := (\text{blocksRead } s \ p) \ (d := \text{blocksRead } s \ p \ d) \\
&\cup \ \{ (\text{block } = \text{disk } s \ d \ p, \ proc = p) \}) \\
&\land \text{inpt } s' = \text{inpt } s \ \land \text{outpt } s' = \text{outpt } s \\
&\land \text{disk } s' = \text{disk } s \ \land \text{phase } s' = \text{phase } s \\
&\land \text{dblock } s' = \text{dblock } s \ \land \text{disksWritten } s' = \text{disksWritten } s
\end{align*}

**definition** EndPhase0 :: state $\Rightarrow$ state $\Rightarrow$ Proc $\Rightarrow$ bool

**where**

\begin{align*}
\text{EndPhase0 } s \ s' \ p = \\
&\ (\text{phase } s \ p = 0) \\
&\land \text{IsMajority } \ (\{ d, \ \text{hasRead } s \ p \ d \}) \\
&\land (\exists b \in \text{Ballot } p. \\
&\ (\forall r \in \text{allBlocksRead } s \ p. \ \text{mbal } r < b) \\
&\land \text{dblock } s' = (\text{dblock } s) \ (p := \\
&\ (\text{SOME } r. \ r \in \text{allBlocksRead } s \ p \\
&\ \land (\forall s \in \text{allBlocksRead } s \ p. \ \text{bal } s \leq \ \text{bal } r)) \ (\vert \text{mbal } := b \vert)) \\
&\land \text{InitializePhase } s \ s' \ p \\
&\land \text{phase } s' = (\text{phase } s) \ (p := 1) \\
&\land \text{inpt } s' = \text{inpt } s \ \land \text{outpt } s' = \text{outpt } s \ \land \text{disk } s' = \text{disk } s)
\end{align*}

**definition** Next :: state $\Rightarrow$ state $\Rightarrow$ bool

**where**

\begin{align*}
\text{Next } s \ s' = (\exists p. \\
&\ \text{StartBallot } s \ s' \ p \\
&\lor (\exists d. \ \text{Phase0Read } s \ s' \ p \ d \\
&\lor \text{Phase1or2Write } s \ s' \ p \ d \\
&\lor (\exists q. q \neq p \ \land \text{Phase1or2Read } s \ s' \ p \ d \ q)) \\
&\lor \text{EndPhase1or2 } s \ s' \ p \\
&\lor \text{Fail } s \ s' \ p \\
&\lor \text{EndPhase0 } s \ s' \ p)
\end{align*}

In the following, for each action or state name we name Hname the corresponding action that includes the history part of the HNext action or state predicate that includes history variables.

**definition** HInit :: state $\Rightarrow$ bool

**where**

\begin{align*}
\text{HInit } s = \\
&\ (\text{Init } s \\
&\land \text{chosen } s = \text{NotAnInput} \\
&\land \text{allInput } s = \text{range } (\text{inpt } s))
\end{align*}
HNextPart is the part of the Next action that is concerned with history variables.

**Definition**

\[ \text{HNextPart} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{bool} \]

where

\[
\text{HNextPart } s \ s' = \\
\quad (\text{chosen } s' = \\
\quad \quad (\text{if } \text{chosen } s \neq \text{NotAnInput} \lor (\forall \ p. \ \text{outpt } s' \ p = \text{NotAnInput} ) \\
\quad \quad \text{then chosen } s \\
\quad \quad \text{else } \text{outpt } s' (\text{SOME } p. \ \text{outpt } s' \ p \neq \text{NotAnInput})) \\
\quad \land \ \text{allInput } s' = \text{allInput } s \cup (\text{range } (\text{inpt } s')))
\]

**Definition**

\[ \text{HNext} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{bool} \]

where

\[
\text{HNext } s \ s' = \\
\quad (\text{Next } s \ s' \\
\quad \land \ \text{HNextPart } s \ s')
\]

We add HNextPart to every action (rather than proving that Next maintains the HInv invariant) to make proofs easier.

**Definition**

\[ \text{HPhase1or2ReadThen} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HPhase1or2ReadThen } s \ s' \ p \ d \ q = (\text{Phase1or2ReadThen } s \ s' \ p \ d \ q \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HEndPhase1} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HEndPhase1 } s \ s' = (\text{EndPhase1 } s \ s' \ p \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HStartBallot} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HStartBallot } s \ s' = (\text{StartBallot } s \ s' \ p \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HPhase1or2Write} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HPhase1or2Write } s \ s' \ p \ d = (\text{Phase1or2Write } s \ s' \ p \ d \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HPhase1or2ReadElse} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HPhase1or2ReadElse } s \ s' \ p \ d \ q = (\text{Phase1or2ReadElse } s \ s' \ p \ d \ q \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HEndPhase2} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HEndPhase2 } s \ s' = (\text{EndPhase2 } s \ s' \ p \land \text{HNextPart } s \ s')
\]

**Definition**

\[ \text{HFail} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]

where

\[
\text{HFail } s \ s' = (\text{Fail } s \ s' \ p \land \text{HNextPart } s \ s')
\]
definition

\text{HPhase0Read} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool}
\text{where}
\text{HPhase0Read} s s' p d = (\text{Phase0Read} s s' p d \land \text{HNextPart} s s')

definition

\text{HEndPhase0} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}
\text{where}
\text{HEndPhase0} s s' p = (\text{EndPhase0} s s' p \land \text{HNextPart} s s')

Since these definitions are the conjunction of two other definitions declaring
them as simplification rules should be harmless.

\text{declare} \text{HPhase1or2ReadThen-def} [\text{simp}]
\text{declare} \text{HPhase1or2ReadElse-def} [\text{simp}]
\text{declare} \text{HEndPhase1-def} [\text{simp}]
\text{declare} \text{HStartBallot-def} [\text{simp}]
\text{declare} \text{HPhase1or2Write-def} [\text{simp}]
\text{declare} \text{HEndPhase2-def} [\text{simp}]
\text{declare} \text{HFail-def} [\text{simp}]
\text{declare} \text{HPhase0Read-def} [\text{simp}]
\text{declare} \text{HEndPhase0-def} [\text{simp}]

end

C Proof of Disk Paxos’ Invariant

theory DiskPaxos-Inv1 imports DiskPaxos-Model begin

C.1 Invariant 1

This is just a type Invariant.

definition \text{Inv1} :: \text{state} \Rightarrow \text{bool}
\text{where}
\text{Inv1} s = (\forall p .
\text{inpt} s p \in \text{Inputs}
\land \text{phase} s p \leq 3
\land \text{finite} (\text{allRdBlks} s p))

definition \text{HInv1} :: \text{state} \Rightarrow \text{bool}
\text{where}
\text{HInv1} s =
(\text{Inv1} s
\land \text{allInput} s \subseteq \text{Inputs})

\text{declare} \text{HInv1-def} [\text{simp}]

We added the assertion that the set allRdBlks is finite for every process p;
one may therefore choose a block with a maximum ballot number in action
EndPhase1.
With the following the lemma, it will be enough to prove Inv1 s’ for every action, without taking the history variables in account.

**Lemma** \( \text{HNextPart-Inv1} \): \( \{ \text{HInv1 s; HNextPart s s’; Inv1 s’} \} \Rightarrow \text{HInv1 s’} \)

by (auto simp add: HNextPart-def Inv1-def)

**Theorem** \( \text{HInit-HInv1} \): \( \text{HInit s} \rightarrow \text{HInv1 s} \)

by (auto simp add: HInit-def Inv1-def Init-def allRdBlks-def)

**Lemma** \( \text{allRdBlks-finite} \):

- **assumes** \( \text{inv: HInv1 s} \)
- **and** \( \text{asm: } \forall p. \text{allRdBlks s’ p } \subseteq \text{insert blk (allRdBlks s p)} \)
- **shows** \( \forall p. \text{finite (allRdBlks s’ p)} \)

**Proof**

fix \( pp \)
from \( inv \)
have \( \forall p. \text{finite (allRdBlks s p)} \)
  by (simp add: Inv1-def)
  hence \( \text{finite (allRdBlks s pp)} \)
  by blast
  with \( asm \)
  show \( \text{finite (allRdBlks s’ pp)} \)
  by (auto intro: finite-subset)
qed

**Theorem** \( \text{HPhase1or2ReadThen-HInv1} \):

- **assumes** \( \text{inv1: HInv1 s} \)
- **and** \( \text{act: HPhase1or2ReadThen s s’ p d q} \)
- **shows** \( \text{HInv1 s’} \)

**Proof**

— we focus on the last conjunct of Inv1
from \( act \)
have \( \forall p. \text{allRdBlks s’ p } \subseteq \text{allRdBlks s p } \cup \{ | \text{block = disk s d q, proc = q} | \} \)
  by (auto simp add: Phase1or2ReadThen-def allRdBlks-def
  split: split-if-asm)
with \( \text{inv1} \)
have \( \forall p. \text{finite (allRdBlks s’ p)} \)
  by (blast dest: allRdBlks-finite)
— the others conjuncts are trivial
with \( \text{inv1 act} \)
show \( ?\text{thesis} \)
  by (auto simp add: Inv1-def Phase1or2ReadThen-def HNextPart-def)
qed

**Theorem** \( \text{HEndPhase1-HInv1} \):

- **assumes** \( \text{inv1: HInv1 s} \)
- **and** \( \text{act: HEndPhase1 s s’ p} \)
- **shows** \( \text{HInv1 s’} \)

**Proof**

from \( \text{inv1 act} \)
have \( \text{Inv1} \ s' \)
by (auto simp add: Inv1-def EndPhase1-def InitializePhase-def allRdBlks-def)
with \( \text{inv1} \) act
show \( \text{thesis} \)
by (auto simp del: HInv1-def dest: HNextPart-Inv1)
qed

theorem HStartBallot-HInv1:
assumes \( \text{inv1} : \text{HInv1} \ s \)
and \( \text{act} : \text{HStartBallot} \ s \ s' \ p \)
shows \( \text{HInv1} \ s' \)
proof
from \( \text{inv1} \) act
have \( \text{Inv1} \ s' \)
by (auto simp add: Inv1-def StartBallot-def InitializePhase-def allRdBlks-def)
with \( \text{inv1} \) act
show \( \text{thesis} \)
by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HPhase1or2Write-HInv1:
assumes \( \text{inv1} : \text{HInv1} \ s \)
and \( \text{act} : \text{HPhase1or2Write} \ s \ s' \ p \ d \)
shows \( \text{HInv1} \ s' \)
proof
from \( \text{inv1} \) act
have \( \text{Inv1} \ s' \)
by (auto simp add: Inv1-def Phase1or2Write-def allRdBlks-def)
with \( \text{inv1} \) act
show \( \text{thesis} \)
by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HPhase1or2ReadElse-HInv1:
assumes \( \text{act} : \text{HPhase1or2ReadElse} \ s \ s' \ p \ d \ q \)
and \( \text{inv1} : \text{HInv1} \ s \)
shows \( \text{HInv1} \ s' \)
using HStartBallot-HInv1[of \( \text{inv1} \) \( \text{act} \)]
by (auto simp add: Phase1or2ReadElse-def)

theorem HEndPhase2-HInv1:
assumes \( \text{inv1} : \text{HInv1} \ s \)
and \( \text{act} : \text{HEndPhase2} \ s \ s' \ p \)
shows \( \text{HInv1} \ s' \)
proof
from \( \text{inv1} \) act
have \( \text{Inv1} \ s' \)
by (auto simp add: Inv1-def EndPhase2-def InitializePhase-def allRdBlks-def)
with \( \text{inv1} \) act
show ?thesis 
  by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HFail-HInv1:
  assumes inv1: HInv1 s
  and act: HFail s s' p
  shows HInv1 s'
proof –
  from inv1 act
  have Inv1 s'
    by (auto simp add: Inv1-def Fail-def InitializePhase-def allRdBlks-def)
  with inv1 act show ?thesis
  by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HPhase0Read-HInv1:
  assumes inv1: HInv1 s
  and act: HPhase0Read s s' p d
  shows HInv1 s'
proof –
  — we focus on the last conjunct of Inv1
  from act
  have ∀ pp. allRdBlks s' pp ⊆ allRdBlks s pp ∪ {⟨block = disk s d p, proc = p⟩}
    by (auto simp add: Phase0Read-def allRdBlks-def
      split: split-if-asm)
  with inv1
  have ∀ p. finite (allRdBlks s' p)
    by (blast dest: allRdBlks-finite)
  — the others conjuncts are trivial
  with inv1 act
  have Inv1 s'
    by (auto simp add: Inv1-def Phase0Read-def)
  with inv1 act
  show ?thesis
    by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HEndPhase0-HInv1:
  assumes inv1: HInv1 s
  and act: HEndPhase0 s s' p
  shows HInv1 s'
proof –
  from inv1 act
  have Inv1 s'
    by (auto simp add: Inv1-def EndPhase0-def allRdBlks-def InitializePhase-def)
  with inv1 act
  show ?thesis
    by (auto simp del: HInv1-def elim: HNextPart-Inv1)

22
HInv1 is an invariant of HNext

**lemma I2a:**

- **assumes** `nxt`: `HNext s s'`
- **and** `inv`: `HInv1 s`
- **shows** `HInv1 s'`

**using** `assms`

**by** `auto`

- `simp add: HNext-def Next-def`
- `auto intro: HStartBallot-HInv1`
- `auto intro: HPhase0Read-HInv1`
- `auto intro: HPhase1or2Write-HInv1`
- `auto simp add: Phase1or2Read-def`
- `intro: HPhase1or2ReadThen-HInv1`
- `HPhase1or2ReadElse-HInv1`
- `auto simp add: EndPhase1or2-def`
- `intro: HEndPhase1-HInv1`
- `HEndPhase2-HInv1`
- `auto intro: HFail-HInv1`
- `auto intro: HEndPhase0-HInv1`

**end**

**theory DiskPaxos-Inv2** **imports** `DiskPaxos-Inv1` **begin**

**C.2 Invariant 2**

The second invariant is split into three main conjuncts called `Inv2a`, `Inv2b`, and `Inv2c`. The main difficulty is in proving the preservation of the first conjunct.

**definition** `rdBy :: state ⇒ Proc ⇒ Proc ⇒ Disk ⇒ BlockProc set` **where**

- `rdBy s p q d = {br . br ∈ blocksRead s q d ∧ proc br = p}`

**definition** `blocksOf :: state ⇒ Proc ⇒ DiskBlock set` **where**

- `blocksOf s p = {dblock s p} ∪ {disk s d p | d . d ∈ UNIV}`
- `∪ {block br | br . br ∈ (UN q d. rdBy s p q d)}`

**definition** `allBlocks :: state ⇒ DiskBlock set`
where allBlocks s = (UN p. blocksOf s p)

**definition Inv2a-innermost :: state ⇒ Proc ⇒ DiskBlock ⇒ bool**

where
Inv2a-innermost s p bk =
  (mbal bk ∈ (Ballot p) ∪ {0})
  ∧ bal bk ∈ (Ballot p) ∪ {0}
  ∧ (bal bk = 0) = (inp bk = NotAnInput)
  ∧ bal bk ≤ mbal bk
  ∧ inp bk ∈ (allInput s) ∪ {NotAnInput})

**definition Inv2a-inner :: state ⇒ Proc ⇒ bool**

where
Inv2a-inner s p = (∀ bk ∈ blocksOf s p. Inv2a-innermost s p bk)

**definition Inv2a :: state ⇒ bool**

where
Inv2a s = (∀ p. Inv2a-inner s p)

**definition Inv2b-inner :: state ⇒ Proc ⇒ Disk ⇒ bool**

where
Inv2b-inner s p d =
  (d ∈ disksWritten s p —>
    (phase s p ∈ {1,2} ∧ disk s d p = dblock s p))
  ∧ (phase s p ∈ {1,2} —>
    (blocksRead s p d ̸= {} —> d ∈ disksWritten s p)
  ∧ ¬ hasRead s p d))

**definition Inv2b :: state ⇒ bool**

where
Inv2b s = (∀ p d. Inv2b-inner s p d)

**definition Inv2c-inner :: state ⇒ Proc ⇒ bool**

where
Inv2c-inner s p =
  ((phase s p = 0 —>
    (dblock s p = InitDB
    ∧ disksWritten s p = {})
    ∧ (∀ d. ∀ br ∈ blocksRead s p d.
      (proc br = p ∧ block br = disk s d p)))
  ∧ (phase s p ̸= 0 —>
    (mbal(dblock s p) ∈ Ballot p
    ∧ bal(dblock s p) ∈ Ballot p ∪ {0}
    ∧ (∀ d. ∀ br ∈ blocksRead s p d.
      mbal(block br) < mbal(dblock s p)))
    ∧ (phase s p ∈ {2,3} —> bal(dblock s p) = mbal(dblock s p))
    ∧ outpt s p = (if phase s p = 3 then inp(dblock s p) else NotAnInput)
    ∧ chosen s ∈ allInput s ∪ {NotAnInput}
    ∧ (∀ p. inp t p ∈ allInput s
      ∧ (chosen s = NotAnInput —> outpt s p = NotAnInput)))))

**definition Inv2c :: state ⇒ bool**
where $\text{Inv2c} s = (\forall p. \text{Inv2c-inner} s p)$

**definition** $\text{HInv2} :: \text{state} \Rightarrow \text{bool}$

where $\text{HInv2} s = (\text{Inv2a} s \land \text{Inv2b} s \land \text{Inv2c} s)$

### C.2.1 Proofs of Invariant 2 a

**theorem** $\text{HInit-Inv2a}$

by (auto simp add: $\text{HInit-def}$ $\text{Init-def}$ $\text{Inv2a-def}$ $\text{Inv2a-inner-def}$ $\text{Inv2a-innermost-def}$ $\text{rdBy-def}$ $\text{blocksOf-def}$ $\text{InitDB-def}$)

For every action we define a action-$\text{blocksOf}$ lemma. We have two cases: either the new $\text{blocksOf}$ is included in the old $\text{blocksOf}$, or the new $\text{blocksOf}$ is included in the old $\text{blocksOf}$ union the new $\text{dblock}$. In the former case the assumption $\text{inv}$ will imply the thesis. In the latter, we just have to prove the innermost predicate for the particular case of the new $\text{dblock}$. This particular case is proved in lemma action-$\text{Inv2a-dblock}$.

**lemma** $\text{HPhase1or2ReadThen-blocksOf}$

by (auto simp add: $\text{Phase1or2ReadThen-def}$ $\text{blocksOf-def}$ $\text{rdBy-def}$)

**theorem** $\text{HPhase1or2ReadThen-Inv2a}$

assumes $\text{inv}$: $\text{Inv2a} s$

shows $\text{Inv2a} s'$

proof (clarsimp simp add: $\text{Inv2a-def}$ $\text{Inv2a-inner-def}$)

fix $pp$ $bk$

assume $bk$: $bk \in \text{blocksOf} s'$ $pp$

with $\text{inv}$ $\text{HPhase1or2ReadThen-blocksOf}[OF \text{act}]$

have $\text{Inv2a-innermost} s' pp bk$

by (auto simp add: $\text{Inv2a-def}$ $\text{Inv2a-inner-def}$)

with $\text{act}$

show $\text{Inv2a-innermost} s' pp bk$

by (auto simp add: $\text{Inv2a-innermost-def}$ $\text{HNextPart-def}$)

qed

**lemma** $\text{InitializePhase-rdBy}$

$\text{InitializePhase} s s' p \Longrightarrow \text{rdBy} s' pp qq dd \subseteq \text{rdBy} s pp qq dd$

by (auto simp add: $\text{InitializePhase-def}$ $\text{rdBy-def}$)

**lemma** $\text{HStartBallot-blocksOf}$

$\text{HStartBallot} s s' p \Longrightarrow \text{blocksOf} s' q \subseteq \text{blocksOf} s q \cup \{\text{dblock} s' q\}$

by (auto simp add: $\text{StartBallot-def}$ $\text{blocksOf-def}$ $\text{dest: subsetD}[OF \text{InitializePhase-rdBy}]$)

**lemma** $\text{HStartBallot-Inv2a-dblock}$

assumes $\text{act}$: $\text{HStartBallot} s s' p$

and $\text{inv2a}$: $\text{Inv2a-innermost} s p \ (\text{dblock} s p)$

25
shows \( \text{Inv2a-innermost } s' \ p \ (\text{dblock } s' \ p) \)

**proof** –

from act
have \( \text{mbal}' : \text{mbal} (\text{dblock } s' \ p) \in \text{Ballot } p \)
  by (auto simp add: \text{StartBallot-def})
from act
have \( \text{bal}' : \text{bal} (\text{dblock } s' \ p) = \text{bal} (\text{dblock } s \ p) \)
  by (auto simp add: \text{StartBallot-def})
with act
have \( \text{inp}' : \text{inp} (\text{dblock } s' \ p) = \text{inp} (\text{dblock } s \ p) \)
  by (auto simp add: \text{StartBallot-def})
from act
have \( \text{mbal} (\text{dblock } s \ p) \leq \text{mbal} (\text{dblock } s' \ p) \)
  by (auto simp add: \text{Inv2a-innermost-def})
with \( \text{bal}' \ \text{inv2a} \)
have \( \text{bal-mbal} : \text{bal} (\text{dblock } s' \ p) \leq \text{mbal} (\text{dblock } s' \ p) \)
  by (auto simp add: \text{Inv2a-innermost-def})
from act
have \( \text{allInput } s \subseteq \text{allInput } s' \)
  by (auto simp add: \text{HNextPart-def} \text{InitializePhase-def} \text{Inv2a-innermost-def})
with \( \text{mbal}' \ \text{bal}' \ \text{inp}' \ \text{bal-mbal} \ \text{act} \ \text{inv2a} \)
show \( ?\text{thesis} \)
by (auto simp add: \text{Inv2a-innermost-def})
qed

**lemma** \( \text{HStartBallot-Inv2a-dblock-q} \):
assumes act: \( \text{HStartBallot } s \ s' \ p \)
and inv2a: \( \text{Inv2a-innermost } s \ q \ (\text{dblock } s \ q) \)
shows \( \text{Inv2a-innermost } s' \ q \ (\text{dblock } s' \ q) \)
**proof**
(cases \( p=q \))
case True
with act inv2a
show \( ?\text{thesis} \)
  by (blast dest: \text{HStartBallot-Inv2a-dblock})
next
case False
with act inv2a
show \( ?\text{thesis} \)
  by (clarsimp simp add: \text{StartBallot-def} \text{HNextPart-def} \text{InitializePhase-def} \text{Inv2a-innermost-def})
qed

**theorem** \( \text{HStartBallot-Inv2a} \):
assumes inv: \( \text{Inv2a } s \)
and act: \( \text{HStartBallot } s \ s' \ p \)
shows \( \text{Inv2a } s' \)
**proof**
(clarsimp simp add: \text{Inv2a-def} \text{Inv2a-inner-def})
fix \( q \ \text{bk} \)
assume \( \text{bk} : \text{bk} \in \text{blocksOf } s' \ q \)
with \( \text{inv} \)
\[
\text{have oldBlks: } bk \in \text{blocksOf } s q \rightarrow \text{Inv2a-innermost } s q bk
\]
\[\text{by (auto simp add: Inv2a-def Inv2a-inner-def)}\]

\[
\text{from } bk \text{ HStartBallot-blocksOf[OF act]
}
\text{have } bk \in \{\text{dblock } s' q\} \cup \text{blocksOf } s q
\]
\[\text{by blast}\]

\[
\text{thus Inv2a-innermost } s' q bk
\]

\text{proof}
\[
\text{assume bk-dblock: } bk \in \{\text{dblock } s' q\}
\]

\[
\text{from } \text{inv}
\text{have inv-q-dblock: Inv2a-innermost } s q (\text{dblock } s q)
\]
\[
\text{by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)}
\]

\[
\text{with } \text{act inv bk-dblock}
\]

\[
\text{show } \text{?thesis}
\]
\[\text{by (blast dest: HStartBallot-Inv2a-dblock-q)}\]

\text{next}
\[
\text{assume bk-in-blocks: } bk \in \text{blocksOf } s q
\]

\[
\text{with oldBlks
}
\text{have Inv2a-innermost } s q bk ..
\]

\[
\text{with } \text{act}
\]

\[
\text{show } \text{?thesis}
\]
\[\text{by (auto simp add: StartBallot-def HNextPart-def InitializePhase-def Inv2a-innermost-def)}\]

\text{qed}

\text{qed}

\text{lemma } HPhase1or2Write-blocksOf:
\[
\begin{align*}
&\text{[ HPhase1or2Write } s s' p d ] \implies \text{blocksOf } s' r \subseteq \text{blocksOf } s r \\
&\text{by (auto simp add: Phase1or2Write-def blocksOf-def rdBby-def)}
\end{align*}
\]

\text{theorem } HPhase1or2Write.Inv2a:
\[
\text{assumes } \text{inv: Inv2a } s
\]

\[
\text{and } \text{act: HPhase1or2Write } s s' p d
\]

\[
\text{shows Inv2a } s'
\]

\text{proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)}
\[
\text{fix } q bk
\]

\[
\text{assume } bk: bk \in \text{blocksOf } s' q
\]

\[
\text{from } \text{inv bk HPhase1or2Write-blocksOf[OF act]
}
\text{have inp-q-bk: Inv2a-innermost } s q bk
\]
\[\text{by (auto simp add: Inv2a-def Inv2a-inner-def)}\]

\[
\text{with } \text{act}
\]

\[
\text{show Inv2a-innermost } s' q bk
\]
\[\text{by (auto simp add: Inv2a-innermost-def HNextPart-def)}\]

\text{qed}

\text{theorem } HPhase1or2ReadElse.Inv2a:
\[
\text{assumes } \text{inv: Inv2a } s
\]

\[
\text{and } \text{act: HPhase1or2ReadElse } s s' p d q
\]

\[
\text{shows Inv2a } s'
\]
proof

from act
have HStartBallot s s' p
  by (simp add: Phase1or2ReadElse-def)
with inv
show ?thesis
  by (auto elim: HStartBallot-Inv2a)
qed

lemma HEndPhase2-blocksOf:
  [ HEndPhase2 s s' p ] \implies \text{blocksOf } q' \subseteq \text{blocksOf } q
by (auto simp add: EndPhase2-def blocksOf-def
  dest: subsetD[OF InitializePhase-rdBy])

theorem HEndPhase2-Inv2a:
  assumes inv: Inv2a s
  and act: HEndPhase2 s s' p
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk \in \text{blocksOf } q'
  from inv bk HEndPhase2-blocksOf[OF act]
  have inp-q-bk: Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act
  show Inv2a-innermost s' q bk
    by (auto simp add: Inv2a-innermost-def HNextPart-def)
qed

lemma HFail-blocksOf:
  HFail s s' p \implies \text{blocksOf } q' \subseteq \text{blocksOf } q \cup \{ \text{dblock } s' q \}
by (auto simp add: Fail-def blocksOf-def
  dest: subsetD[OF InitializePhase-rdBy])

lemma HFail-Inv2a-dblock-q:
  assumes act: HFail s s' p
  and inv: Inv2a-innermost s q (dblock s q)
  shows Inv2a-innermost s' q (dblock s' q)
proof (cases p=q)
  case True
  with act
  have dblock s' q = InitDB
    by (simp add: Fail-def)
  with True
  show ?thesis
    by (auto simp add: InitDB-def Inv2a-innermost-def)
next
case False
with inv act
show \( ?\text{thesis} \)
\[
\text{by (auto simp add: Fail-def HNextPart-def InitializePhase-def Inv2a-innermost-def)}
\]
qed

theorem \( H\text{Fail-Inv2a} \):
assumes \( \text{inv: Inv2a } s \)
and \( \text{act: HFail } s \ s' \ p \)
shows \( \text{Inv2a } s' \)
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
fix \( q \ bk \)
assume \( bk: bk \in \text{blocksOf } s' \ q \)
with \( H\text{Fail-blocksOf}[OF \text{act}] \)
have \( \text{dblock-blocks: } bk \in \{ \text{dblock } s' \ q \} \cup \text{blocksOf } s \ q \)
by blast
thus \( \text{Inv2a-innermost } s' \ q \ bk \)
proof
assume \( bk\text{-dblock: } bk \in \{ \text{dblock } s' \ q \} \)
from \( \text{inv} \)
have \( \text{inv-q-dbloc}: \text{Inv2a-innermost } s \ q \ (\text{dblock } s \ q) \)
by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
with \( \text{act} \ bk\text{-dblock} \)
show \( ?\text{thesis} \)
by (blast dest: HFail-Inv2a-dbloc-q)
next
assume \( bk\text{-in-blocks: } bk \in \text{blocksOf } s \ q \)
with \( \text{inv} \)
have \( \text{Inv2a-innermost } s \ q \ bk \)
by (auto simp add: Inv2a-def Inv2a-inner-def)
with \( \text{act} \)
show \( ?\text{thesis} \)
by (auto simp add: Fail-def HNextPart-def InitializePhase-def Inv2a-innermost-def)
qued
qed

lemma \( H\text{Phase0Read-blocksOf} \):
\( H\text{Phase0Read } s \ s' \ p \ d \implies \text{blocksOf } s' \ q \subseteq \text{blocksOf } s \ q \)
by (auto simp add: Phase0Read-def InitializePhase-def blocksOf-def rdBy-def)

theorem \( H\text{Phase0Read-Inv2a} \):
assumes \( \text{inv: Inv2a } s \)
and \( \text{act: HPhase0Read } s \ s' \ p \ d \)
shows \( \text{Inv2a } s' \)
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
fix \( q \ bk \)
assume \( bk: bk \in \text{blocksOf } s' \ q \)
from \( \text{inv} \ bk \ H\text{Phase0Read-blocksOf}[OF \text{act}] \)
have inp-q-bk: Inv2a-innermost s q bk
  by (auto simp add: Inv2a-def Inv2a-inner-def)
with act
show Inv2a-innermost s' q bk
  by (auto simp add: Inv2a-innermost-def HNextPart-def)
qed

lemma HEndPhase0-blocksOf:
  HEndPhase0 s s' p \implies\ blocksOf s' q \subseteq\ blocksOf s q \cup\ \{dblock s' q\}
by (auto simp add: EndPhase0-def blocksOf-def
   dest: subsetD[OF InitializePhase-rdBy])

lemma HEndPhase0-blocksRead:
  assumes act: HEndPhase0 s s' p
  shows \exists \ d. \ blocksRead s p d \neq\ {}
proof -
  from act
  have IsMajority({d. hasRead s p d p}) by (simp add: EndPhase0-def)
  hence {d. hasRead s p d p} \neq\ {} by (rule majority-nonempty)
  thus ?thesis
  by (auto simp add: hasRead-def)
qed

EndPhase0 has the additional difficulty of having a choose expression. We prove that there exists an \( x \) such that the predicate of the choose expression holds, and then apply someI: \( \exists \ ?P \ ?x \implies\ ?P \ (Eps \ ?P) \).

lemma HEndPhase0-some:
  assumes act: HEndPhase0 s s' p
  and inv1: Inv1 s
  shows \( (\exists b. b \in allBlocksRead s p \land (\forall t \in allBlocksRead s p. bal t \leq bal b)) \land (\forall t \in allBlocksRead s p. \ (\forall t \in allBlocksRead s p. \ bal t \leq bal (SOME b. b \in allBlocksRead s p \land (\forall t \in allBlocksRead s p. bal t \leq bal b)))) \)
proof -
  from inv1 have finite (bal ' allBlocksRead s p) (is finite ?S)
    by (simp add: Inv1-def allBlocksRead-def)
  moreover
  from HEndPhase0-blocksRead[OF act]
  have \( \{\} \neq\ {} \)
    by (auto simp add: allBlocksRead-def allRdBlks-def)
  ultimately
  have Max ?S \in\ ?S \ and \ (\forall t \in ?S. t \leq Max ?S) \ by\ auto
  hence \( \exists \ r. \ (\forall t \in ?S. t \leq r) \land (\forall t \in allBlocksRead s p. \bal t \leq bal mblk) \)
    by (auto simp add: hasRead-def)
  then obtain mblk
    where \( mblk \in allBlocksRead s p \land (\forall t \in allBlocksRead s p. \bal t \leq bal mblk) \) (is ?P mblk)
by auto
thus ?thesis
  by (rule someI)
qed

lemma HEndPhase0-dblock-allBlocksRead:
  assumes act: HEndPhase0 s s' p
  and inv1: Inv1 s
  shows dblock s' p ∈ (λx. x (mbal := mbal(dblock s' p))) · allBlocksRead s p
using act HEndPhase0-some[OF act inv1]
  by(auto simp add: EndPhase0-def)

lemma HNextPart-allInput-or-NotAnInput:
  assumes act: HNextPart s s' p
  and inv2a: Inv2a-innermost s p (dblock s' p)
  shows inp (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
proof –
  from act
  have allInput s' = allInput s ∪ (range (inpt s'))
    by(simp add: HNextPart-def)
  moreover
  from inv2a
  have inp (dblock s' p) ∈ allInput s ∪ {NotAnInput}
    by(simp add: Inv2a-innermost-def)
  ultimately show ?thesis
    by blast
qed

lemma HEndPhase0-Inv2a-allBlocksRead:
  assumes act: HEndPhase0 s s' p
  and inv2a: Inv2a-inner s p
  and inv2c: Inv2c-inner s p
  shows ∀ t ∈ (λx. x (mbal := mbal (dblock s' p))) · allBlocksRead s p.
    Inv2a-innermost s p t
proof –
  from act
  have mbal': mbal (dblock s' p) ∈ Ballot p
    by(auto simp add: EndPhase0-def)
  from inv2c act
  have allproc-p: ∀ d. ∀ br ∈ blocksRead s p d. proc br = p
    by(simp add: Inv2c-inner-def EndPhase0-def)
  with inv2a
  have allBlocks-inv2a: ∀ t ∈ allBlocksRead s p. Inv2a-innermost s p t
proof(auto simp add: Inv2a-inner-def allBlocksRead-def
  allRdBlks-def blocksOf-def rdBy-def)
  fix d bk
  assume bk-in-blocksRead: bk ∈ blocksRead s p d
  and inv2a-bk: ∀ x ∈ {u. ∃ d. u = disk s d p}
    ∪ {block br | br. (∃ q d. br ∈ blocksRead s q d)
\( \land \text{proc br} = p \}\). Inv2a-innermost s p x

with allproc-p have proc bk = p by auto

with bk-in-blocksRead inv2a-bk
show Inv2a-innermost s p (block bk) by blast

qed

from act
have mbal'\(-gt\): \( \forall bk \in \text{allBlocksRead s p}. \ \text{mbal bk} \leq \text{mbal (dblock s' p)} \)
by(auto simp add: EndPhase0-def)
with mbal' allBlocks-inv2a
show \(?thesis
proof (auto simp add: Inv2a-innermost-def)

fix t
assume t-blocksRead: \( t \in \text{allBlocksRead s p} \)
with allBlocks-inv2a
have bal t \leq mbal t by (auto simp add: Inv2a-innermost-def)
moreover
from t-blocksRead mbal'\(-gt\)
have mbal t \leq mbal (dblock s' p) by blast
ultimately show bal t \leq mbal (dblock s' p)
by auto

qed

qed

lemma HEndPhase0-Inv2a-dblock:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and inv2a: Inv2a-inner s p
and inv2c: Inv2c-inner s p
shows Inv2a-innermost s' p (dblock s' p)

proof –

from act inv2a inv2c
have t1: \( \forall t \in (\lambda x. x (|mbal:= \text{mbal (dblock s' p)}|)) \cdot \text{allBlocksRead s p} \)
Inv2a-innermost s p t
by(blast dest: HEndPhase0-Inv2a-allBlocksRead)
from act inv1
have dblock s' p \in (\lambda x. x (|mbal:= \text{mbal (dblock s' p)}|)) \cdot \text{allBlocksRead s p}
by(simp, blast dest: HEndPhase0-dblock-allBlocksRead)
with t1
have inv2-dblock: Inv2a-innermost s p (dblock s' p) by auto
with act
have inp (dblock s' p) \in allInput s' \cup \{NotAnInput\}
by(auto dest: HNextPart-allInput-or-NotAnInput)
with inv2-dblock
show \(?thesis
by(auto simp add: Inv2a-innermost-def)

qed

lemma HEndPhase0-Inv2a-dblock-q:
assumes act: HEndPhase0 s s' p
and \( inv1: Inv1 s \)
and \( inv2a: Inv2a-inner s q \)
and \( inv2c: Inv2c-inner s p \)
shows \( Inv2a-innermost s' q (dblock s' q) \)

proof (cases \( p=q \))
  case True
with act inv2a inv2c inv1
show ?thesis
  by (blast dest: HEndPhase0-Inv2a-dblock)
next
  case False
from inv2a
have inv-q-dblock: \( Inv2a-innermost s q (dblock s q) \)
  by (auto simp add: Inv2a-inner-def blocksOf-def)
with False act
show ?thesis
  by (clarsimp simp add: EndPhase0-def HNextPart-def InitializePhase-def Inv2a-innermost-def)

qed

theorem HEndPhase0-Inv2a:
assumes inv: \( Inv2a s \)
and act: \( HEndPhase0 s s' p \)
and inv1: \( Inv1 s \)
and inv2c: \( Inv2c-inner s p \)
shows \( Inv2a s' \)

proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
fix q bk
assume bk: \( bk \in \text{blocksOf } s' q \)
with HEndPhase0-blocksOf[OF act]
have dblock-blocks: \( bk \in \{dblock s' q\} \cup \text{blocksOf } s q \)
  by blast
thus \( Inv2a-innermost s' q bk \)

proof
  from inv
  have inv-q: \( Inv2a-inner s q \)
    by (auto simp add: Inv2a-def)
  assume bk \( \in \{dblock s' q\} \)
  with act inv1 inv2c inv-q
  show ?thesis
    by (blast dest: HEndPhase0-Inv2a-dblock-q)
next
  assume bk-in-blocks: \( bk \in \text{blocksOf } s q \)
  with inv
  have Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act show ?thesis
    by (auto simp add: EndPhase0-def HNextPart-def InitializePhase-def Inv2a-innermost-def)
qed

lemma \texttt{HEndPhase1-blocksOf}:
\[ \text{HEndPhase1 } s \; s' \; p \implies \text{blocksOf } s' \; q \subseteq \text{blocksOf } s \; q \cup \{ \text{dblock } s' \; q \} \]
by (auto simp add: \texttt{EndPhase1-def blocksOf-def dest: \texttt{subsetD[OF InitializePhase-rdBy]})

lemma \texttt{maxBlk-in-nonInitBlks}:
assumes \( b : \; b \in \text{nonInitBlks } s \; p \)
and \texttt{inv1: Inv1 } s
shows \( \text{maxBlk } s \; p \in \text{nonInitBlks } s \; p \)
\[ \land (\forall c \in \text{nonInitBlks } s \; p. \; \text{bal } c \leq \text{bal } (\text{maxBlk } s \; p)) \]
proof –
have \( \text{nibals-finite: finite } (\text{bal }' \; (\text{nonInitBlks } s \; p)) \) \( \text{(is finite } \; ?S) \)
proof (rule finite-imageI)
from \texttt{inv1}
have \( \text{finite } (\text{allRdBlks } s \; p) \)
by (auto simp add: \texttt{Inv1-def})
thus \( \text{finite } (\text{allBlocksRead } s \; p) \)
by (auto simp add: \texttt{allBlocksRead-def})
thus \( \text{finite } (\text{blocksSeen } s \; p) \)
by (simp add: \texttt{blocksSeen-def})
thus \( \text{finite } (\text{nonInitBlks } s \; p) \)
by(auto simp add: \texttt{nonInitBlks-def intro: finite-subset})
qed
from \( b \) have \( \text{bal }' \; \text{nonInitBlks } s \; p \neq \{\} \)
by auto
with \( \text{nibals-finite} \)
have \( \text{Max } ?S \in ?S \; \text{and } \forall bb \in ?S. \; bb \leq \text{Max } ?S \) by auto
hence \( \exists \; mb \in ?S. \; \forall bb \in ?S. \; bb \leq \text{mb } . . \)
then obtain \texttt{mbblk}
where \( \; \text{mbblk } \in \text{nonInitBlks } s \; p \)
\[ \land (\forall c \in \text{nonInitBlks } s \; p. \; \text{bal } c \leq \text{bal }\text{mbblk}) \]
(is \( ?P \; \text{mbblk})
by auto
hence \( ?P \; (\text{SOME } b. \; ?P b) \)
by (rule someI)
thus \( ?\text{thesis} \)
by (simp add: \texttt{maxBlk-def})
qed

lemma \texttt{blocksOf-nonInitBlks}:
\[ (\forall p \; bk. \; bk \in \text{blocksOf } s \; p \implies P \; bk) \]
\[ \implies bk \in \text{nonInitBlks } s \; p \implies P \; bk \]
by (auto simp add: \texttt{allRdBlks-def blocksOf-def nonInitBlks-def blocksSeen-def allBlocksRead-def rdBy-def, blast})
lemma maxBlk-allInput:
assumes inv: Inv2a s
and mblk: maxBlk s p ∈ nonInitBlks s p
shows inp (maxBlk s p) ∈ allInput s
proof –
from inv
have blocks: ∀ p bk. bk ∈ blocksOf s p
→ inp bk ∈ (allInput s) ∪ {NotAnInput}
  by(auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def)
from mblk NotAnInput
have inp (maxBlk s p) ≠ NotAnInput
  by(auto simp add: nonInitBlks-def)
with mblk blocksOf-nonInitBlks[OF blocks]
show ?thesis
  by auto
qed

lemma HEndPhase1-dblock-allInput:
assumes act: HEndPhase1 s s′ p
and inv1: HInv1 s
and inv2: Inv2a s
shows inp′: inp (dblock s′ p) ∈ allInput s′
proof –
from act
have inpt: inpt s p ∈ allInput s′
  by(auto simp add: HNextPart-def EndPhase1-def)
have nonInitBlks s p ≠ {} → inp (maxBlk s p) ∈ allInput s
proof
  assume ni: nonInitBlks s p ≠ {}
  with inv1
  have maxBlk s p ∈ nonInitBlks s p
    by(auto simp add: HInv1-def maxBlk-in-nonInitBlks)
  with inv2
  show inp (maxBlk s p) ∈ allInput s
    by(blast dest: maxBlk-allInput)
qed
with act inpt
show ?thesis
  by(auto simp add: EndPhase1-def HNextPart-def)
qed

lemma HEndPhase1-Inv2a-dblock:
assumes act: HEndPhase1 s s′ p
and inv1: HInv1 s
and inv2: Inv2a s
and inv2c: Inv2c-inner s p
shows Inv2a-innermost s′ p (dblock s′ p)
proof –
from inv1 act have inv1′: HInv1 s′
by (blast dest: HEndPhase1-HInv1)
from inv2
have inv2a: Inv2a-innermost s p (dblock s p)
  by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
from act inv2c
have mbal’: mbal (dblock s’ p) ∈ Ballot p
  by (auto simp add: EndPhase1-def Inv2c-def Inv2c-inner-def)
moreover
from act
have bal’: bal (dblock s’ p) = mbal (dblock s p)
  by (auto simp add: EndPhase1-def)
moreover
from act inv2c
have inp’: inp (dblock s’ p) ∈ allInput s’
  by (blast dest: HEndPhase1-dblock-allInput)
moreover
with inv1’ NotAnInput
have inp (dblock s’ p) ≠ NotAnInput
  by (auto simp add: HInv1-def)
ultimately show ?thesis
  using act inv2a
  by (auto simp add: Inv2a-innermost-def EndPhase1-def)
qed

lemma HEndPhase1-Inv2a-dblock-q:
  assumes act: HEndPhase1 s s’ p
  and inv1: HInv1 s
  and inv: Inv2a s
  and inv2c: Inv2c-inner s p
  shows Inv2a-innermost s’ q (dblock s’ q)
proof (cases p=q)
  case True
  with act inv inv2c inv1
  show ?thesis
    by (blast dest: HEndPhase1-Inv2a-dblock)
next
  case False
  from inv
  have inv-q-dblock: Inv2a-innermost s q (dblock s q)
    by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
  with False act
  show ?thesis
    by (clarsimp simp add: EndPhase1-def HNextPart-def
      InitializePhase-def Inv2a-innermost-def)
qed

theorem HEndPhase1-Inv2a:
  assumes act: HEndPhase1 s s’ p
  and inv1: HInv1 s
and \( inv: \text{Inv2a} \ s \)
and \( inv2c: \text{Inv2c-inner} \ s \ p \)
shows \( \text{Inv2a} \ s' \)
proof (clarsimp simp add: \text{Inv2a-def} \text{Inv2a-inner-def})
fix \( q \ bk \)
assume \( bk\text{-in-bks}: bk \in \text{blocksOf} \ s' \ q \)
with \( \text{HEndPhase1-blocksOf[OF act]} \)
have \( \text{dblock-blocks}: bk \in \{ \text{dblock} \ s' \ q \} \cup \text{blocksOf} \ s \ q \)
by blast
thus \( \text{Inv2a-innermost} \ s' \ q \ bk \)
proof
assume \( bk \in \{ \text{dblock} \ s' \ q \} \)
with \( \text{act inv1 inv2c inv} \)
show \?thesis
by (blast dest: \text{HEndPhase1-Inv2a-dblock-q})
next
assume \( bk\text{-in-blocks}: bk \in \text{blocksOf} \ s \ q \)
with \( \text{inv} \)
have \( \text{Inv2a-innermost} \ s \ q \ bk \)
by (auto simp add: \text{Inv2a-def} \text{Inv2a-inner-def})
with \( \text{act} \)
show \?thesis
by (auto simp add: \text{EndPhase1-def} \text{HNextPart-def} \text{InitializePhase-def} \text{Inv2a-innermost-def})
qed
qed

C.2.2 Proofs of Invariant 2 b

Invariant 2b is proved automatically, given that we expand the definitions involved.

theorem \text{HInit-Inv2b}: \text{HInit} \ s \ \\
\rightarrow \text{Inv2b} \ s
by (auto simp add: \text{HInit-def} \text{Init-def} \text{Inv2b-def} \text{Inv2b-inner-def} \text{InitDB-def})

theorem \text{HPhase1or2ReadThen-Inv2b}:
[ Inv2b \ s; \text{HPhase1or2ReadThen} \ s \ s' \ p \ d \ q ] \\
\implies \text{Inv2b} \ s'
by (auto simp add: \text{Phase1or2ReadThen-def} \text{Inv2b-def} \text{Inv2b-inner-def} \text{hasRead-def})

theorem \text{HStartBallot-Inv2b}:
[ Inv2b \ s; \text{HStartBallot} \ s \ s' \ p ] \\
\implies \text{Inv2b} \ s'
by (auto simp add: \text{StartBallot-def} \text{InitializePhase-def} \text{Inv2b-def} \text{Inv2b-inner-def} \text{hasRead-def})

theorem \text{HPhase1or2Write-Inv2b}:
[ Inv2b \ s; \text{HPhase1or2Write} \ s \ s' \ p \ d ] \\
\implies \text{Inv2b} \ s'

by (auto simp add: Phase1or2Write-def Inv2b-def
Inv2b-inner-def hasRead-def)

theorem HPhase1or2ReadElse-Inv2b:
  [ Inv2b s; HPhase1or2ReadElse s s' p d q ]
  \Rightarrow Inv2b s'
by (auto simp add: Phase1or2ReadElse-def StartBallot-def hasRead-def
InitializePhase-def Inv2b-def Inv2b-inner-def)

theorem HEndPhase1-Inv2b:
  [ Inv2b s; HEndPhase1 s s' p ] \Rightarrow Inv2b s'
by (auto simp add: EndPhase1-def InitializePhase-def
Inv2b-def Inv2b-inner-def hasRead-def)

theorem HFail-Inv2b:
  [ Inv2b s; HFail s s' p ] \Rightarrow Inv2b s'
by (auto simp add: Fail-def InitializePhase-def
Inv2b-def Inv2b-inner-def hasRead-def)

theorem HEndPhase2-Inv2b:
  [ Inv2b s; HEndPhase2 s s' p ] \Rightarrow Inv2b s'
by (auto simp add: EndPhase2-def InitializePhase-def
Inv2b-def Inv2b-inner-def hasRead-def)

theorem HPhase0Read-Inv2b:
  [ Inv2b s; HPhase0Read s s' p d ] \Rightarrow Inv2b s'
by (auto simp add: Phase0Read-def Inv2b-def
Inv2b-inner-def hasRead-def)

theorem HEndPhase0-Inv2b:
  [ Inv2b s; HEndPhase0 s s' p ] \Rightarrow Inv2b s'
by (auto simp add: EndPhase0-def InitializePhase-def
Inv2b-def Inv2b-inner-def hasRead-def)

C.2.3 Proofs of Invariant 2 c

theorem HInit-Inv2c: HInit s \rightarrow Inv2c s
by (auto simp add: HInit-def Init-def Inv2c-def Inv2c-inner-def)

lemma HNextPart-Inv2c-chosen:
  assumes hnp: HNextPart s s'
  and inv2c: Inv2c s
  and outpt': \forall p. outpt s' p = (if phase s' p = 3
then inp(dblock s' p) else NotAnInput)
  and inp-dblk: \forall p. inp (dblock s' p) \in allInput s' \cup \{NotAnInput\}
  shows chosen s' \in allInput s' \cup \{NotAnInput\}
proof (auto simp add: HNextPart-inner-def split: split-if-asm)

qed

lemma HNextPart-chosen:
  assumes hnp: HNextPart s s'
  shows chosen s' = NotAnInput ⟷ (∀ p. outpt s' p = NotAnInput)
using hnp
proof (auto simp add: HNextPart-def split: split-if-asm)
  fix p pa
  assume o1: outpt s' p ≠ NotAnInput
  and o2: outpt s' (SOME p. outpt s' p ≠ NotAnInput) = NotAnInput
from o1 have ∃ p. outpt s' p ≠ NotAnInput
    by auto
  hence outpt s' (SOME p. outpt s' p ≠ NotAnInput) ≠ NotAnInput
    by (rule someI-ex)
  with o2
  show outpt s' pa = NotAnInput
    by simp
qed

lemma HNextPart-allInput:
  [ HNextPart s s', Inv2c s ] ⟷ ∀ p. inp s' p ∈ allInput s'
by (auto simp add: HNextPart-def Inv2c-def Inv2c-inner-def)

theorem HPhase1or2ReadThen-Inv2c:
  assumes inv: Inv2c s
  and act: HPhase1or2ReadThen s s' p d q
  and inv2a: Inv2a s
  shows Inv2c s'
proof
  from inv2a act
  have inv2a': Inv2a s'
    by (blast dest: HPhase1or2ReadThen-Inv2a)
  from act inv
  have outpt': ∀ p. outpt s' p = (if phase s' p = 3
    then inp (dblock s' p)
    else NotAnInput)
    by (auto simp add: Phase1or2ReadThen-def Inv2c-def Inv2c-inner-def)
  from inv2a'
  have dblk': ∀ p. inp (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
    by (auto simp add: Inv2a-def Inv2a-inner-def
      Inv2a-innermost-def blocksOf-def)
  with act inv outpt'
  have chosen': chosen s' ∈ allInput s' ∪ {NotAnInput}
    by (auto dest: HNextPart-Inv2c-chosen)
  from act inv

  39
have \(\forall p. \text{ inp } s' p \in \text{ allInput } s'\)
\(\wedge (\text{ chosen } s' = \text{ NotAnInput} \rightarrow \text{ outpt } s' p = \text{ NotAnInput})\)
by(auto dest: HNextPart-chosen HNextPart-allInput)

with outpt' chosen' act inv
show ?thesis
by(auto simp add: Phase1or2ReadThen-def Inv2c-def Inv2c-inner-def)

qed

theorem HStartBallot-Inv2c:
assumes inv: Inv2c s
and act: HStartBallot s s' p
and inv2a: Inv2a s
shows Inv2c s'

proof –
from act
have phase': phase s' p = 1
by(simp add: StartBallot-def)

from act
have phase: phase s p \in \{1, 2\}
by(simp add: StartBallot-def)

from act inv
have mbal': mbal(dblock s' p) \in \text{ Ballot } p
by(auto simp add: StartBallot-def Inv2c-def Inv2c-inner-def)

from inv phase
have bal(dblock s p) \in \text{ Ballot } p \cup \{0\}
by(auto simp add: Inv2c-def Inv2c-inner-def)

with act
have bal': bal(dblock s' p) \in \text{ Ballot } p \cup \{0\}
by(auto simp add: StartBallot-def)

from act inv phase phase'
have blks': (\forall d. \forall br \in \text{ blocksRead } s' p d.
\quad \text{mbal(block br) < mbal(dblock s' p))})
by(auto simp add: StartBallot-def InitializePhase-def
Inv2c-def Inv2c-inner-def)

from inv2a act
have inv2a': Inv2a s'
by(blast dest: HStartBallot-Inv2a)

from act inv
have outpt': \(\forall p. \text{ outpt } s' p = (\text{ if } \text{ phase } s' p = 3 \)
then inp(dblock s' p)
else \text{ NotAnInput})\)
by(auto simp add: StartBallot-def Inv2c-def Inv2c-inner-def)

from inv2a'
have dblk: \(\forall p. \text{ inp } (\text{ dblock } s' p) \in \text{ allInput } s' \cup \{\text{NotAnInput}\})\)
by(auto simp add: Inv2a-def Inv2a-inner-def
Inv2a-innermost-def blocksOf-def)

with act inv outpt'
have chosen': chosen s' \in \text{ allInput } s' \cup \{\text{NotAnInput}\}
by(auto dest: HNextPart-Inv2c-chosen)
from act inv
have allinp: \(\forall p. \text{inpt } s' p \in \text{allInput } s'\)
\[\text{\wedge (chosen } s' = \text{NotAnInput } \rightarrow \text{outpt } s' p = \text{NotAnInput})\]
by(auto dest: HNextPart-chosen HNextPart-allInput)
with phase' mbal' bal' outpt' chosen' act inv blks'
show \(?\text{thesis}\)
by(auto simp add: StartBallot-def InitializePhase-def
Inv2c-def Inv2c-inner-def)
qed

**theorem** HPhase1or2Write-Inv2c:
assumes inv: Inv2c s
and act: HPhase1or2Write s s' p d
and inv2a: Inv2a s
shows Inv2c s'
proof –
from inv2a act
have inv2a': Inv2a s'
by(blast dest: HPhase1or2Write-Inv2a)
from act inv
have outpt': \(\forall p. \text{outpt } s' p = (\text{if phase } s' p = 3\)
\[\text{then inp(dblock } s' p)\]
\[\text{else NotAnInput})\]
by(auto simp add: Phase1or2Write-def Inv2c-def Inv2c-inner-def)
from inv2a'
have dblk: \(\forall p. \text{inp (dblock } s' p) \in \text{allInput } s' \cup \{\text{NotAnInput}\}\)
by(auto simp add: Inv2a-def Inv2a-inner-def
Inv2a-innermost-def blocksOf-def)
with act inv outpt'
have chosen': chosen s' \in \text{allInput } s' \cup \{\text{NotAnInput}\}
by(auto dest: HNextPart-Inv2c-chosen)
from act inv
have allinp: \(\forall p. \text{inpt } s' p \in \text{allInput } s' \wedge (\text{chosen } s' = \text{NotAnInput } \rightarrow \text{outpt } s' p = \text{NotAnInput})\)
by(auto dest: HNextPart-chosen HNextPart-allInput)
with outpt' chosen' act inv
show \(?\text{thesis}\)
by(auto simp add: Phase1or2Write-def Inv2c-def Inv2c-inner-def)
qed

**theorem** HPhase1or2ReadElse-Inv2c:
[ Inv2c s; HPhase1or2ReadElse s s' p d q; Inv2a s ] \(\Rightarrow\) Inv2c s'
by(auto simp add: Phase1or2ReadElse-def dest: HStartBallot-Inv2c)

**theorem** HEndPhase1-Inv2c:
assumes inv: Inv2c s
and act: HEndPhase1 s s' p
and inv2a: Inv2a s
and \( inv1 : HInv1 \)
shows \( Inv2c \)
proof –
  from \( inv \)
  have \( Inv2c-inner \ s \ p \) by (auto simp add: \( Inv2c-def \))
  with \( inv2a \ act inv1 \)
  have \( inv2a' : Inv2a \ s' \)
    by (blast dest: \( HEndPhase1-Inv2a \))
  from \( act inv \)
  have \( mbal' : mbal(dblock \ s' \ p) \in \) Ballot \( p \)
    by (auto simp add: \( EndPhase1-def Inv2c-def Inv2c-inner-def \))
  from \( act \)
  have \( bal' : bal(dblock \ s' \ p) = mbal(dblock \ s' \ p) \)
    by (auto simp add: \( EndPhase1-def \))
  from \( act inv \)
  have \( blks' : (\forall d. \forall br \in \) blocksRead \( s' \ p d \).
    \( mbal(block \ br) < mbal(dblock \ s' \ p) \))
    by (auto simp add: \( EndPhase1-def InitializePhase-def \)
      \( Inv2c-def Inv2c-inner-def \))
  from \( act inv \)
  have \( outpt' : \forall p. outpt \ s' \ p = (if \) phase \( s' \ p = 3 \then
    \) inp(dblock \ s' \ p) \else \) NotAnInput \)
    by (auto simp add: \( EndPhase1-def Inv2c-def Inv2c-inner-def \))
  from \( inv2a' \)
  have \( dblk : \forall p. \) inp(dblock \ s' \ p) \in allInput \( s' \cup \) \{NotAnInput\}
    by (auto simp add: \( Inv2a-def Inv2a-inner-def \)
      \( Inv2a-innermost-def blocksOf-def \))
  with \( act \) \( inv \) \( outpt' \)
  have \( chosen' : chosen \ s' \in allInput \ s' \cup \) \{NotAnInput\}
    by (auto dest: \( HNextPart-Inv2c-chosen \))
  from \( act inv \)
  have \( allinp : \forall p. \) inp \( s' \ p \in allInput \ s' \)
    \( \land \) (chosen \( s' = NotAnInput \)
    \( \mapsto \) outpt \( s' \ p = NotAnInput \))
    by (auto dest: \( HNextPart-chosen HNextPart-allInput \))
  with \( mbal' bal' blks' outpt' chosen' act inv \)
  show \?thesis
    by (auto simp add: \( EndPhase1-def InitializePhase-def \)
      \( Inv2c-def Inv2c-inner-def \))
qed

theorem \( HEndPhase2-Inv2c \):
  assumes \( inv: Inv2c \ s \)
  and \( act: HEndPhase2 \ s \ s' \ p \)
  and \( inv2a: Inv2a \ s \)
  shows \( Inv2c \ s' \)
proof –
  from \( inv2a \) \( act \)
have \( \text{inv2a'}: \text{Inv2a s'} \)
  by (blast dest: HEndPhase2-Inv2a)

from act inv
have \( \text{outpt'}: \forall p. \text{outpt s'} p = (\text{if phase s'} p = 3 \\
  \text{then inp(dblock s'} p) \\
  \text{else NotAnInput}) \)
  by (auto simp add: EndPhase2-def Inv2c-def Inv2c-inner-def)

from \( \text{inv2a'} \)
have \( \text{dblk}: \forall p. \text{inp(dblock s'} p) \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)

with act inv outpt
have \( \text{chosen}: \text{chosen s'} \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto dest: HNextPart-Inv2c-chosen)

from act inv
have \( \text{allinp}: \forall p. \text{inpt s'} p \in \text{allInput s'} \\
  \land (\text{chosen s'} = \text{NotAnInput} \\
  \rightarrow \text{outpt s'} p = \text{NotAnInput}) \)
  by (auto dest: HNextPart-chosen HNextPart-allInput)

show \( ?\text{thesis} \)
  by (auto simp add: EndPhase2-def InitializePhase-def 
  Inv2c-def Inv2c-inner-def)

qed

theorem HFail-Inv2c:
  assumes \( \text{inv}: \text{Inv2c s} \)
  and \( \text{act}: \text{HFail s s'} p \)
  and \( \text{inv2a}: \text{Inv2a s} \)
  shows \( \text{Inv2c s'} \)

proof —
  from \( \text{inv2a act} \)
  have \( \text{inv2a'}: \text{Inv2a s'} \)
    by (blast dest: HFail-Inv2a)

from act inv
have \( \text{outpt'}: \forall p. \text{outpt s'} p = (\text{if phase s'} p = 3 \\
  \text{then inp(dblock s'} p) \\
  \text{else NotAnInput}) \)
  by (auto simp add: Fail-def Inv2c-def Inv2c-inner-def)

from \( \text{inv2a'} \)
have \( \text{dblk}: \forall p. \text{inp(dblock s'} p) \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto simp add: Inv2a-def Inv2a-inner-def 
  Inv2a-innermost-def blocksOf-def)

with act inv outpt
have \( \text{chosen}: \text{chosen s'} \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto dest: HNextPart-Inv2c-chosen)

from act inv
have \( \text{allinp}: \forall p. \text{inpt s'} p \in \text{allInput s'} \\
  \land (\text{chosen s'} = \text{NotAnInput} \\
  \rightarrow \text{outpt s'} p = \text{NotAnInput}) \)
by (auto dest: HNextPart-chosen HNextPart-allInput)

with outpt' chosen' act inv

show ?thesis
  by (auto simp add: Fail-def InitializePhase-def
             Inv2c-def Inv2c-inner-def)

qed

theorem HPhase0Read-Inv2c:
  assumes inv: Inv2c s
  and act: HPhase0Read s s' p d
  and inv2a: Inv2a s
  shows Inv2c s'

proof
  from inv2a act
  have inv2a': Inv2a s'
    by (blast dest: HPhase0Read-Inv2a)
  from act inv
  have outpt': \( \forall p. \text{outpt } s' p = (\text{if } \text{phase } s' p = 3 \quad \text{then } \text{inp}(\text{dblock } s' p) \quad \text{else } \text{NotAnInput}) \)
    by (auto simp add: Phase0Read-def Inv2c-def Inv2c-inner-def)
  from inv2a'
  have dblk: \( \forall p. \text{inp}(\text{dblock } s' p) \in \text{allInput } s' \cup \{ \text{NotAnInput} \} \)
    by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
  with act inv outpt'
  have chosen': chosen' s' \in \text{allInput } s' \cup \{ \text{NotAnInput} \}
    by (auto dest: HNextPart-Inv2c-chosen)
  from act inv
  have allinp: \( \forall p. \text{inpt } s' p \in \text{allInput } s' \)
    \( \land (\text{chosen } s' = \text{NotAnInput} \quad \rightarrow \text{outpt } s' p = \text{NotAnInput}) \)
    by (auto dest: HNextPart-chosen HNextPart-allInput)
  with outpt' chosen' act inv
  show ?thesis
    by (auto simp add: Phase0Read-def
             Inv2c-def Inv2c-inner-def)
qed

theorem HEndPhase0-Inv2c:
  assumes inv: Inv2c s
  and act: HEndPhase0 s s' p
  and inv2a: Inv2a s
  and inv1: Inv1 s
  shows Inv2c s'

proof
  from inv
  have Inv2c-inner s p by (auto simp add: Inv2c-def)
  with inv2a act inv1

qed
have \( \text{inv2a}' : \text{Inv2a} s' \)
by (blast dest: \text{HEndPhase0-Inv2a})
hence \( \text{bal}' : \text{bal}(\text{dblock} s' p) \in \text{Ballot} p \cup \{0\} \)
by (auto simp add: \text{Inv2c-def}\ \text{Inv2c-inner-def}\ \text{blocksOf-def})

from act inv
have \( \text{mbal}' : \text{mbal}(\text{dblock} s' p) \in \text{Ballot} p \)
by (auto simp add: \text{EndPhase0-def}\ \text{InitializePhase-def}\ \text{Inv2c-def}\ \text{Inv2c-inner-def})

from act inv
have \( \text{outpt}' : \forall p. \text{outpt} s' p = (\text{if phase} s' p = 3 \then \text{inp}(\text{dblock} s' p) \else \text{NotAnInput}) \)
by (auto simp add: \text{EndPhase0-def}\ \text{Inv2c-def}\ \text{Inv2c-inner-def})

from \( \text{inv2a}' \)
have \( \text{dblk} : \forall p. \text{inp}(\text{dblock} s' p) \in \text{allInput} s' \cup \{\text{NotAnInput}\} \)
by (auto simp add: \text{Inv2c-def}\ \text{Inv2c-inner-def}\ \text{blocksOf-def})

with act inv \( \text{outpt}' \)
have \( \text{chosen}' : \forall p. \text{inpt} s' p \in \text{allInput} s' \land (\text{chosen} s' = \text{NotAnInput} \implies \text{outpt} s' p = \text{NotAnInput}) \)
by (auto dest: \text{HNextPart-Inv2c-chosen})

from act inv
have \( \text{allinp} : \forall p. \text{inpt} s' p \in \text{allInput} s' \land (\text{chosen} s' = \text{NotAnInput} \implies \text{outpt} s' p = \text{NotAnInput}) \)
by (auto dest: \text{HNextPart-chosen}\ \text{HNextPart-allInput})

with \( \text{mbal}' \ \text{bal}' \ \text{blks}' \ \text{outpt}' \ \text{chosen}' \ \text{act inv} \)

show \( ?\text{thesis} \)
by (auto simp add: \text{EndPhase0-def}\ \text{InitializePhase-def}\ \text{Inv2c-def}\ \text{Inv2c-inner-def})

qed

theorem \( \text{HInit-HInv2} : \)
\( \text{HInit} s \implies \text{HInv2} s \)
using \( \text{HInit-Inv2a} \ \text{HInit-Inv2b} \ \text{HInit-Inv2c} \)
by (auto simp add: \text{HInv2-def})

\( \text{HInv1} \land \text{HInv2} \) is an invariant of \( \text{HNext} \).

lemma \( I2b : \)
assumes \( \text{nxt} : \text{HNext} s s' \)
and \( \text{inv} : \text{HInv1} s \land \text{HInv2} s \)
shows \( \text{HInv2} s' \)
proof (auto simp add: \text{HInv2-def})
show \( \text{Inv2a} s' \) using \( \text{assms} \)
by (auto simp add: \text{HInv2-def}\ \text{HNext-def}\ \text{Next-def},
auto intro: \text{HStartBallot-Inv2a},

45
theory DiskPaxos-Inv3 imports DiskPaxos-Inv2 begin

C.3 Invariant 3

This invariant says that if two processes have read each other’s block from disk $d$ during their current phases, then at least one of them has read the other’s current block.

definition HInv3-L :: state ⇒ Proc ⇒ Proc ⇒ Disk ⇒ bool where
\begin{align*}
Hinv3-L \quad s \quad p \quad q \quad d &= \quad (\text{phase} \quad s \quad p \in \{1,2\} \\
& \quad \wedge \quad \text{phase} \quad s \quad q \in \{1,2\} \\
& \quad \wedge \quad \text{hasRead} \quad s \quad p \quad d \quad q \\
& \quad \wedge \quad \text{hasRead} \quad s \quad q \quad d \quad p) \\
\text{definition} \quad Hinv3-R \quad :: \quad \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool} \\
\text{where} \quad Hinv3-R \quad s \quad p \quad q \quad d &= \quad (\{(\text{block} \quad = \quad \text{dblock} \quad s \quad q, \quad \text{proc} \quad = \quad q)\} \in \text{blocksRead} \quad s \quad p \quad d \\
& \quad \vee \quad \{(\text{block} \quad = \quad \text{dblock} \quad s \quad p, \quad \text{proc} \quad = \quad p)\} \in \text{blocksRead} \quad s \quad q \quad d) \\
\text{definition} \quad Hinv3-inner \quad :: \quad \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool} \\
\text{where} \quad Hinv3-inner \quad s \quad p \quad q \quad d &= \quad (Hinv3-L \quad s \quad p \quad q \quad d \rightarrow Hinv3-R \quad s \quad p \quad q \quad d) \\
\text{definition} \quad Hinv3 \quad :: \quad \text{state} \Rightarrow \text{bool} \\
\text{where} \quad Hinv3 \quad s &= \quad (\forall \quad p \quad q \quad d. \quad Hinv3-inner \quad s \quad p \quad q \quad d) \\
\end{align*}

C.3.1 Proofs of Invariant 3

\text{theorem} \quad HInit-Hinv3: \quad HInit \quad s \quad \Rightarrow \quad Hinv3 \quad s \\
\text{by simp add: HInit-def Init-def Hinv3-def} \\
\text{Hinv3-inner-def Hinv3-L-def Hinv3-R-def) }

\text{lemma} \quad InitPhase-Hinv3-p: \\
[ \text{InitPhase} \quad s \quad s' \quad p; \quad Hinv3-L \quad s' \quad p \quad q \quad d ] \quad \Rightarrow \quad Hinv3-R \quad s' \quad p \quad q \quad d \\
\text{by (auto simp add: InitPhase-def Hinv3-inner-def} \\
\text{hasRead-def Hinv3-L-def Hinv3-R-def) }

\text{lemma} \quad InitPhase-Hinv3-q: \\
[ \text{InitPhase} \quad s \quad s' \quad q; \quad Hinv3-L \quad s' \quad p \quad q \quad d ] \quad \Rightarrow \quad Hinv3-R \quad s' \quad p \quad q \quad d \\
\text{by (auto simp add: InitPhase-def Hinv3-inner-def} \\
\text{hasRead-def Hinv3-L-def Hinv3-R-def) }

\text{lemma} \quad Hinv3-L-sym: \quad Hinv3-L \quad s \quad p \quad q \quad d \quad \Rightarrow \quad Hinv3-L \quad s \quad q \quad p \quad d \\
\text{by (auto simp add: Hinv3-L-def) }

\text{lemma} \quad Hinv3-R-sym: \quad Hinv3-R \quad s \quad p \quad q \quad d \quad \Rightarrow \quad Hinv3-R \quad s \quad q \quad p \quad d \\
\text{by (auto simp add: Hinv3-R-def) }

\text{lemma} \quad Phase1or2ReadThen-Hinv3-pq: \\
\text{assumes act: Phase1or2ReadThen} \quad s \quad s' \quad p \quad d \quad q \\
\text{and inv-L\': Hinv3-L} \quad s' \quad p \quad q \quad d \\
\text{and pq: } \quad p \neq q \\
\text{and inv2b: Inv2b} \quad s \\
\text{shows } \quad Hinv3-R \quad s' \quad p \quad q \quad d \\
\text{proof } - \\
\text{from inv-L\' act pq} \\
\text{have phase} \quad s \quad q \in \{1,2\} \wedge \text{hasRead} \quad s \quad q \quad d \quad p \\
\text{by (auto simp add: Phase1or2ReadThen-def Hinv3-L-def} \\
\text{hasRead-def split: split-if-asmp) }

47
with inv2b
have disk s d q = dblock s q
  by (auto simp add: Inv2b-def Inv2b-inner-def
       hasRead-def)
with act
show ?thesis
  by (auto simp add: Phase1or2ReadThen-def HInv3-def
       HInv3-inner-def HInv3-R-def)
qed

lemma Phase1or2ReadThen-HInv3-hasRead:
[ [ ¬ hasRead s pp dd qq;
   Phase1or2ReadThen s s' p d q;
   pp≠p ∨ qq≠q ∨ dd≠d ]
  ⇒ ¬ hasRead s' pp dd qq
by (auto simp add: hasRead-def Phase1or2ReadThen-def)

theorem HPhase1or2ReadThen-HInv3:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv3 s
  and pq: p≠q
  and inv2b: Inv2b s
  shows HInv3 s'
proof (clarsimp simp add: HInv3-def HInv3-inner-def)
  fix pp qq dd
  assume h3l': HInv3-L s' pp qq dd
  show HInv3-R s' pp qq dd
  proof (cases HInv3-L s pp qq dd)
    case True
    with inv
    have HInv3-R s pp qq dd
      by (auto simp add: HInv3-def HInv3-inner-def)
    with act h3l'
    show ?thesis
      by (auto simp add: HInv3-R-def HInv3-L-def
                         Phase1or2ReadThen-def)
  next
    case False
    assume nh3l: ¬ HInv3-L s pp qq dd
    show HInv3-R s' pp qq dd
    proof (cases ((pp=p ∧ qq=q) ∨ (pp=q ∧ qq=p)) ∧ dd=d)
      case True
      with act pq inv2b h3l' HInv3-L-sym[OF h3l']
      show ?thesis
        by (auto dest: Phase1or2ReadThen-HInv3-pq HInv3-R-sym)
    next
    case False
    from nh3l h3l' act
    have (¬hasRead s pp dd ∨ ¬hasRead s qq dd pp)
\[\land \text{hasRead} \ s' \ pp \ dd \ qq \ \land \text{hasRead} \ s' \ qq \ dd \ pp\]
\[\text{by (auto simp add: HInv3-L-def Phase1or2ReadThen-def) with act False show } ?\text{thesis by (auto dest: Phase1or2ReadThen-HInv3-hasRead)}\]
qed
qed
qed

\text{lemma }
\text{StartBallot-HInv3-p:}[\begin{array}{l}
\text{StartBallot} \ s \ s' \ p \ ; \ HInv3-L \ s' \ p \ q \ d \\
\end{array}]
\[\Rightarrow \ HInv3-R \ s' \ p \ q \ d\]
\[\text{by (auto simp add: StartBallot-def dest: InitPhase-HInv3-p)}\]

\text{lemma }
\text{StartBallot-HInv3-q:}[\begin{array}{l}
\text{StartBallot} \ s \ s' \ q \ ; \ HInv3-L \ s' \ p \ q \ d \\
\end{array}]
\[\Rightarrow \ HInv3-R \ s' \ p \ q \ d\]
\[\text{by (auto simp add: StartBallot-def dest: InitPhase-HInv3-q)}\]

\text{lemma }
\text{StartBallot-HInv3-nL:}[\begin{array}{l}
\text{StartBallot} \ s \ s' \ t \ ; \ \neg HInv3-L \ s \ p \ q \ d \ ; \ t \neq p \ ; \ t \neq q \\
\end{array}]
\[\Rightarrow \ \neg HInv3-L \ s' \ p \ q \ d\]
\[\text{by (auto simp add: StartBallot-def InitializePhase-def HInv3-L-def hasRead-def)}\]

\text{lemma }
\text{StartBallot-HInv3-R:}[\begin{array}{l}
\text{StartBallot} \ s \ s' \ t \ ; \ HInv3-R \ s \ p \ q \ d \ ; \ t \neq p \ ; \ t \neq q \\
\end{array}]
\[\Rightarrow \ HInv3-R \ s' \ p \ q \ d\]
\[\text{by (auto simp add: StartBallot-def InitializePhase-def HInv3-R-def hasRead-def)}\]

\text{lemma }
\text{StartBallot-HInv3-t:}[\begin{array}{l}
\text{StartBallot} \ s \ s' \ t \ ; \ HInv3-inner \ s \ p \ q \ d \ ; \ t \neq p \ ; \ t \neq q \\
\end{array}]
\[\Rightarrow \ HInv3-inner \ s' \ p \ q \ d\]
\[\text{by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-nL StartBallot-HInv3-R)}\]

\text{lemma }
\text{StartBallot-HInv3:}
\[\begin{array}{l}
\text{assumes act: StartBallot} \ s \ s' \ t \\
\text{and inv: HInv3-inner \ s \ p \ q \ d} \\
\text{shows HInv3-inner \ s' \ p \ q \ d} \\
\end{array}\]
\[\text{proof (cases } t=p \lor t=q)\]
\[\text{case True}\]
\[\text{with act inv}\]
\[\text{show } ?\text{thesis by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-p StartBallot-HInv3-q)}\]
\[\text{next}\]
\[\text{case False}\]
with inv act
show ?thesis
by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-t)
qed

theorem HStartBallot-HInv3:
[ HStartBallot s s' p; HInv3 s ] =⇒ HInv3 s'
by (auto simp add: Hinv3-def dest: StartBallot-HInv3)

theorem HPhase1or2ReadElse-HInv3:
[ HPhase1or2ReadElse s s' p d q; HInv3 s ] =⇒ HInv3 s'
by (auto simp add: Phase1or2ReadElse-def HInv3-def
dest: StartBallot-HInv3)

theorem HPhase1or2Write-HInv3:
assumes act: HPhase1or2Write s s' p d
and inv: HInv3 s
shows HInv3 s'
proof (auto simp add: HInv3-def)
fix pp qq dd
show HInv3-inner s' pp qq dd
proof (cases HInv3-L s pp qq dd)
case True
with inv
have HInv3-R s pp qq dd
by (simp add: HInv3-def HInv3-inner-def)
with act
show ?thesis
by (auto simp add: HInv3-inner-def HInv3-R-def
Phase1or2Write-def)
next
case False
with act
have ¬HInv3-L s' pp qq dd
by (auto simp add: HInv3-L-def hasRead-def Phase1or2Write-def)
thus ?thesis
by (simp add: HInv3-inner-def)
qed
qed

lemma EndPhase1-HInv3-p:
[ EndPhase1 s s' p; HInv3-L s' p q d ] =⇒ HInv3-R s' p q d
by (auto simp add: EndPhase1-def dest: InitPhase-HInv3-p)

lemma EndPhase1-HInv3-q:
[ EndPhase1 s s' q; HInv3-L s' p q d ] =⇒ HInv3-R s' p q d
by (auto simp add: EndPhase1-def dest: InitPhase-HInv3-q)

lemma EndPhase1-HInv3-nL:
lemma EndPhase1-HInv3-R:
\[
\begin{array}{l}
\text{EndPhase1 } s s' t; \neg \text{HInv3-L } s p q d; t \neq p; t \neq q,
\hline
\quad \Rightarrow \text{HInv3-R } s' p q d
\end{array}
\]
by(auto simp add: EndPhase1-def InitializePhase-def
\hspace{1cm} HInv3-L-def hasRead-def)

lemma EndPhase1-HInv3-t:
\[
\begin{array}{l}
\text{EndPhase1 } s s' t; \text{HInv3-inner } s p q d; t \neq p; t \neq q,
\hline
\quad \Rightarrow \text{HInv3-inner } s' p q d
\end{array}
\]
by(auto simp add: HInv3-inner-def dest: EndPhase1-HInv3-nL
\hspace{1cm} EndPhase1-HInv3-R)

lemma EndPhase1-HInv3:
assumes act: EndPhase1 s s' t
and inv: HInv3-inner s p q d
shows HInv3-inner s' p q d
proof(cases t=p ∨ t=q)
then case True
with act inv
show ?thesis
  by(auto simp add: HInv3-inner-def
\hspace{1cm} dest: EndPhase1-HInv3-p EndPhase1-HInv3-q)
next
case False
with inv act
show ?thesis
  by(auto simp add: HInv3-inner-def dest: EndPhase1-HInv3-t)
qed

theorem HEndPhase1-HInv3:
\[
\begin{array}{l}
\text{HEndPhase1 } s s' p; \text{HInv3 } s
\hline
\quad \Rightarrow \text{HInv3 } s'
\end{array}
\]
by(auto simp add: HInv3-def dest: EndPhase1-HInv3)

lemma EndPhase2-HInv3-p:
\[
\begin{array}{l}
\text{EndPhase2 } s s' p; \text{HInv3-L } s' p q d
\hline
\quad \Rightarrow \text{HInv3-R } s' p q d
\end{array}
\]
by(auto simp add: HInv3-def dest: InitPhase-HInv3-p)

lemma EndPhase2-HInv3-q:
\[
\begin{array}{l}
\text{EndPhase2 } s s' q; \text{HInv3-L } s' p q d
\hline
\quad \Rightarrow \text{HInv3-R } s' p q d
\end{array}
\]
by(auto simp add: HInv3-def dest: InitPhase-HInv3-q)

lemma EndPhase2-HInv3-nL:
\[
\begin{array}{l}
\text{EndPhase2 } s s' t; \neg \text{HInv3-L } s p q d; t \neq p; t \neq q,
\hline
\quad \Rightarrow \neg \text{HInv3-L } s' p q d
\end{array}
\]
by (auto simp add: EndPhase2-def InitializePhase-def
     HInv3-L-def hasRead-def)

lemma EndPhase2-HInv3-R:
[ EndPhase2 s s' t; HInv3-R s p q d; t≠p; t≠ q ]
⇒ HInv3-R s' p q d
by (auto simp add: EndPhase2-def InitializePhase-def
     HInv3-R-def hasRead-def)

lemma EndPhase2-HInv3-t:
[ EndPhase2 s s' t; HInv3-inner s p q d; t≠p; t≠ q ]
⇒ HInv3-inner s' p q d
by (auto simp add: HInv3-inner-def
     dest: EndPhase2-HInv3-nL EndPhase2-HInv3-R)

lemma EndPhase2-HInv3:
assumes act: EndPhase2 s s' t
and inv: HInv3-inner s p q d
shows HInv3-inner s' p q d
proof (cases t=p ∨ t=q)
case True
  with act inv
  show ?thesis
    by (auto simp add: HInv3-inner-def
        dest: EndPhase2-HInv3-p EndPhase2-HInv3-q)
next
  case False
  with inv act
  show ?thesis
    by (auto simp add: HInv3-inner-def dest: EndPhase2-HInv3-t)
qed

theorem HEndPhase2-HInv3:
[ HEndPhase2 s s' p; HInv3 s ]⇒ HInv3 s'
by (auto simp add: HInv3-def dest: EndPhase2-HInv3)

lemma Fail-HInv3-p:
[ Fail s s' p; HInv3-L s' p q d ]⇒ HInv3-R s' p q d
by (auto simp add: Fail-def dest: InitPhase-HInv3-p)

lemma Fail-HInv3-q:
[ Fail s s' q; HInv3-L s' p q d ]⇒ HInv3-R s' p q d
by (auto simp add: Fail-def dest: InitPhase-HInv3-q)

lemma Fail-HInv3-nL:
[ Fail s s' t; ¬HInv3-L s p q d; t≠p; t≠ q ]
⇒ ¬HInv3-L s' p q d
by (auto simp add: Fail-def InitializePhase-def
     HInv3-L-def hasRead-def)
lemma Fail-HInv3-R:
\[
[ \text{Fail } s s' t; \text{HInv3-R } s p q d; t \neq p; t \neq q ] \\
\implies \text{HInv3-R } s' p q d
\]
by(auto simp add: Fail-def InitializePhase-def HInv3-R-def hasRead-def)

lemma Fail-HInv3-t:
\[
[ \text{Fail } s s' t; \text{HInv3-inner } s p q d; t \neq p; t \neq q ] \\
\implies \text{HInv3-inner } s' p q d
\]
by(auto simp add: HInv3-inner-def dest: Fail-HInv3-nL Fail-HInv3-R)

lemma Fail-HInv3:
assumes act: \text{Fail } s s' t \\
and inv: \text{HInv3-inner } s p q d \\
shows \text{HInv3-inner } s' p q d
proof(cases \( t = p \lor t = q \))
  case True
  with inv act
  show \(?thesis\) \\
  by(auto simp add: HInv3-inner-def dest: Fail-HInv3-p Fail-HInv3-q)
next
  case False
  with inv act
  show \(?thesis\) \\
  by(auto simp add: HInv3-inner-def dest: Fail-HInv3-t)
qed

theorem HFail-HInv3:
\[
[ \text{HFail } s s' p; \text{HInv3 } s ] \\
\implies \text{HInv3 } s'
\]
by(auto simp add: HInv3-def dest: Fail-HInv3)

theorem HPhase0Read-HInv3:
assumes act: \text{HPhase0Read } s s' p d \\
and inv: \text{HInv3 } s \\
shows \text{HInv3 } s'
proof(auto simp add: HInv3-def)
  fix pp qq dd
  show \text{HInv3-inner } s' pp qq dd \\
  proof(cases \text{HInv3-L } s pp qq dd)
    case True
    with inv 
    have \text{HInv3-R } s pp qq dd \\
    by(simp add: HInv3-def HInv3-inner-def)
    with act
    show \(?thesis\) \\
    by(auto simp add: HInv3-inner-def HInv3-R-def Phase0Read-def)
  qed
qed
next
  case False
  with act
  have \neg HInv3-L s' pp qq dd
    by (auto simp add: HInv3-L-def hasRead-def Phase0Read-def)
  thus ?thesis
    by (simp add: HInv3-inner-def)
qed

lemma EndPhase0-HInv3-p:
[ [ EndPhase0 s s' p; HInv3-L s' p q d ] ]
\Rightarrow HInv3-R s' p q d
by (auto simp add: EndPhase0-def dest: InitPhase-HInv3-p)

lemma EndPhase0-HInv3-q:
[ [ EndPhase0 s s' q; HInv3-L s' p q d ] ]
\Rightarrow HInv3-R s' p q d
by (auto simp add: EndPhase0-def dest: InitPhase-HInv3-q)

lemma EndPhase0-HInv3-nL:
[ [ EndPhase0 s s' t; \neg HInv3-L s p q d; t\neq p; t\neq q ] ]
\Rightarrow \neg HInv3-L s' p q d
by (auto simp add: EndPhase0-def InitializePhase-def
    HInv3-L-def hasRead-def)

lemma EndPhase0-HInv3-R:
[ [ EndPhase0 s s' t; HInv3-R s p q d; t\neq p; t\neq q ] ]
\Rightarrow HInv3-R s' p q d
by (auto simp add: EndPhase0-def InitializePhase-def
    HInv3-R-def hasRead-def)

lemma EndPhase0-HInv3-t:
[ [ EndPhase0 s s' t; HInv3-inner s p q d; t\neq p; t\neq q ] ]
\Rightarrow HInv3-inner s' p q d
by (auto simp add: HInv3-inner-def
    dest: EndPhase0-HInv3-nL EndPhase0-HInv3-R)

lemma EndPhase0-HInv3:
  assumes act: "EndPhase0 s s' t"
  and inv: "HInv3-inner s p q d"
  shows "HInv3-inner s' p q d"
proof (cases t=p \or t=q)
  case True
  with act inv
  show ?thesis
    by (auto simp add: HInv3-inner-def
        dest: EndPhase0-HInv3-p EndPhase0-HInv3-q)
next
case False
with inv act
show ?thesis
  by (auto simp add: HInv3-inner-def dest: EndPhase0-HInv3-t)
qed

theorem HEndPhase0-HInv3:
[ HEndPhase0 s s' p; HInv3 s ] \implies HInv3 s'
by (auto simp add: HInv3-def dest: EndPhase0-HInv3)

HInv1 \land HInv2 \land HInv3 is an invariant of HNext.

lemma I2c:
  assumes nxt: HNext s s'
  and inv: HInv1 s \land HInv2 s \land HInv3 s
  shows HInv3 s' using assms
  by (auto simp add: HNext-def Next-def,
       auto intro: HStartBallot-HInv3,
       auto intro: HPhase0Read-HInv3,
       auto intro: HPhase1or2Write-HInv3,
       auto simp add: Phase1or2Read-def HInv2-def
              intro: HPhase1or2ReadThen-HInv3
                       HPhase1or2ReadElse-HInv3,
       auto simp add: EndPhase1or2-def
              intro: HEndPhase1-HInv3
                       HEndPhase2-HInv3,
       auto intro: HFail-HInv3,
       auto intro: HEndPhase0-HInv3)
end

theory DiskPaxos-Inv4 imports DiskPaxos-Inv2 begin

C.4 Invariant 4

This invariant expresses relations among mbal and bal values of a processor and of its disk blocks. HInv4a asserts that, when p is not recovering from a failure, its mbal value is at least as large as the bal field of any of its blocks, and at least as large as the mbal field of its block on some disk in any majority set. HInv4b conjunct asserts that, in phase 1, its mbal value is actually greater than the bal field of any of its blocks. HInv4c asserts that, in phase 2, its bal value is the mbal field of all its blocks on some majority set of disks. HInv4d asserts that the bal field of any of its blocks is at most as large as the mbal field of all its disk blocks on some majority set of disks.

definition MajoritySet :: Disk set set
  where MajoritySet = { D. IsMajority(D) }

definition HInv4a1 :: state \Rightarrow Proc \Rightarrow bool
where $HInv4a1 \; s \; p = (\forall \; bk \in \text{blocksOf } s \; p. \; \text{bal} \; bk \; \leq \; \text{mbal} \; (\text{dblock } s \; p))$

definition $HInv4a2 :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$
where
$HInv4a2 \; s \; p = (\forall \; D \in \text{MajoritySet} \; (\exists \; d \in D. \; \text{mbal} \; (\text{disk } s \; d \; p) \leq \; \text{mbal} \; (\text{dblock } s \; p))$
\land \; \text{bal} \; (\text{disk } s \; d \; p) \leq \; \text{bal} \; (\text{dblock } s \; p))))$

definition $HInv4a :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$
where
$HInv4a \; s \; p = (\text{phase } s \; p \; \neq \; 0 \; \rightarrow \; HInv4a1 \; s \; p \; \land \; HInv4a2 \; s \; p)$

definition $HInv4b :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$
where
$HInv4b \; s \; p = (\text{phase } s \; p \; = \; 1 \; \rightarrow \; (\forall \; bk \in \text{blocksOf } s \; p. \; \text{bal} \; bk \; < \; \text{mbal} \; (\text{dblock } s \; p)))$

definition $HInv4c :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$
where
$HInv4c \; s \; p = (\text{phase } s \; p \; \in \{2, 3\} \; \rightarrow \; (\exists \; D \in \text{MajoritySet}. \; \forall \; d \in D. \; \text{mbal} \; (\text{disk } s \; d \; p) = \; \text{bal} \; (\text{dblock } s \; p)))$

definition $HInv4d :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$
where
$HInv4d \; s \; p = (\forall \; bk \in \text{blocksOf } s \; p. \; \exists \; D \in \text{MajoritySet}. \; \forall \; d \in D. \; \text{bal} \; bk \; \leq \; \text{mbal} \; (\text{disk } s \; d \; p))$

definition $HInv4 :: \text{state} \Rightarrow \text{bool}$
where
$HInv4 \; s \; = \; (\forall \; p. \; HInv4a \; s \; p \; \land \; HInv4b \; s \; p \; \land \; HInv4c \; s \; p \; \land \; HInv4d \; s \; p)$

The initial state implies Invariant 4.

theorem $HInit\Rightarrow HInv4 :: HInit \; s \; \Rightarrow \; HInv4 \; s$
using $\text{Disk-isMajority}$
by (auto simp add: $HInit$-def $Init$-def $HInv4$-def $HInv4a$-def $HInv4a1$-def
\$HInv4a2$-def $HInv4b$-def $HInv4c$-def $HInv4d$-def $\text{MajoritySet}$-def $\text{blocksOf}$-def $\text{InitDB}$-def $\text{rdBy}$-def)

To prove that the actions preserve $HInv4$, we do it for one conjunct at a time.

For each action $\text{action}ss'q$ and conjunct $x \in a, b, c, d$ of $HInv4xs'p$, we prove two lemmas. The first lemma $\text{action}$-$HInv4x$-$p$ proves the case of $p = q$, while lemma $\text{action}$-$HInv4x$-$q$ proves the other case.

C.4.1 Proofs of Invariant 4a

lemma $HStartBallot\Rightarrow HInv4a1$: 
assumes $\text{act} :: HStartBallot \; s \; s' \; p$
and $\text{inv} :: HInv4a1 \; s \; p$
and $\text{inv2a} :: HInv2a-inner \; s' \; p$
shows $HInv4a1 \; s' \; p$
proof (auto simp add: $HInv4a1$-def)
fix $bk$

56
assume $bk \in \text{blocksOf } s' p$
with $\text{HStartBallot} - \text{blocksOf}$[OF act]
have $bk \in \{\text{dblock } s' p\} \cup \text{blocksOf } s p$
by blast
thus $\text{bal} bk \leq \text{mbal} (\text{dblock } s' p)$
proof
assume $bk \in \{\text{dblock } s' p\}$
with $\text{inv2a}$
show $\text{thesis}$
by(auto simp add: $\text{Inv2a-innermost-def} \text{Inv2a-inner-def} \text{blocksOf-def}$)
next
assume $bk \in \text{blocksOf } s p$
with $\text{inv act}$
show $\text{thesis}$
by(auto simp add: $\text{StartBallot-def} \text{HInv4a1-def}$)
qed

lemma $\text{HStartBallot-HInv4a2}$:
assumes $\text{act: HStartBallot } s s' p$
and $\text{inv: HInv4a } s p$
shows $\text{HInv4a2 } s' p$
proof(auto simp add: $\text{HInv4a2-def}$)
fix $D$
assume $\text{Dmaj: } D \in \text{MajoritySet}$
from $\text{inv Dmaj}$
have $\exists d \in D. \ {\text{mbal} (\text{disk } s d p) \leq \text{mbal} (\text{dblock } s p)}$
$\land \ {\text{bal} (\text{disk } s d p) \leq \text{bal} (\text{dblock } s p)}$
by(auto simp add: $\text{HInv4a2-def}$)
then obtain $d$
where $d \in D$
$\land \ {\text{mbal} (\text{disk } s d p) \leq \text{mbal} (\text{dblock } s p)}$
$\land \ {\text{bal} (\text{disk } s d p) \leq \text{bal} (\text{dblock } s p)}$
by auto
with $\text{act}$
have $d \in D$
$\land \ {\text{mbal} (\text{disk } s' d p) \leq \text{mbal} (\text{dblock } s' p)}$
$\land \ {\text{bal} (\text{disk } s' d p) \leq \text{bal} (\text{dblock } s' p)}$
by(auto simp add: $\text{StartBallot-def}$)
with $\text{Dmaj}$
show $\exists d \in D. \ {\text{mbal} (\text{disk } s' d p) \leq \text{mbal} (\text{dblock } s' p)}$
$\land \ {\text{bal} (\text{disk } s' d p) \leq \text{bal} (\text{dblock } s' p)}$
by auto
qed

lemma $\text{HStartBallot-HInv4a-p}$:
assumes $\text{act: HStartBallot } s s' p$
and $\text{inv: HInv4a } s p$
and $\text{inv2a: Inv2a-inner } s' p$
57
shows $H_{inv4a} s' p$
using act inv inv2a
proof
from act have phase: $0 < phase s p$
  by (auto simp add: StartBallot-def)
from act inv inv2a show ?thesis
  by (auto simp del: HStartBallot-def simp add: HInv4a-def phase
       elim: HStartBallot-HInv4a1 HStartBallot-HInv4a2)
qed

lemma HStartBallot-HInv4a-q:
assumes act: $H_{startBallot} s s' p$
and inv: $H_{inv4a} s q$
and pnq: $p \neq q$
shows $H_{inv4a} s' q$
proof
from act pnq have blocksOf $s' q \subseteq$ blocksOf $s q$
  by (auto simp add: StartBallot-def InitializePhase-def
        blocksOf-def rdBy-def)
with act inv pnq show ?thesis
  by (auto simp add: StartBallot-def HInv4a-def
               HInv4a1-def HInv4a2-def)
qed

theorem HStartBallot-HInv4a:
assumes act: $H_{startBallot} s s' p$
and inv: $H_{inv4a} s q$
and inv2a: $Inv2a s'$
shows $H_{inv4a} s' q$
proof (cases $p = q$)
case True
from inv2a have $Inv2a-inner s' p$
  by (auto simp add: Inv2a-def)
with act inv True show ?thesis
  by (blast dest: HStartBallot-HInv4a-p)
next
case False
with act inv show ?thesis
  by (blast dest: HStartBallot-HInv4a-q)
qed

lemma Phase1or2Write-HInv4a1:
Phase1or2Write $s s' p d ; \ HInv^4a1 s q \implies HInv^4a1 s' q$
by(auto simp add: Phase1or2Write-def HInv^4a1-def blocksOf-def rdBy-def)

lemma Phase1or2Write-HInv^4a2:
[ Phase1or2Write $s s' p d ; \ HInv^4a2 s q \implies HInv^4a2 s' q $]
by(auto simp add: Phase1or2Write-def HInv^4a2-def)

theorem HPhase1or2Write-HInv^4a:
assumes act: HPhase1or2Write $s s' p d$
and inv: $HInv^4a s q$
shows $HInv^4a s' q$
proof –
  from act
  have phase': phase $s = phase s'$
    by(simp add: Phase1or2Write-def)
  show ?thesis
  proof(cases phase $s q = 0$)
    case True
    with phase' act
    show ?thesis
    by(auto simp add: HInv^4a-def)
  next
    case False
    with phase' act inv
    show ?thesis
    by(auto simp add: HInv^4a-def dest: Phase1or2Write-HInv^4a1 Phase1or2Write-HInv^4a2)
  qed
qed

lemma HPhase1or2ReadThen-HInv^4a1-p:
assumes act: HPhase1or2ReadThen $s s' p d q$
and inv: $HInv^4a1 s p$
shows $HInv^4a1 s' p$
proof(auto simp: HInv^4a1-def)
  fix bk
  assume bk: bk $\in$ blocksOf $s' p$
  with HPhase1or2ReadThen-blocksOf[OF act]
  have bk $\in$ blocksOf $s p$ by auto
  with inv act
  show bal bk $\leq$ mbal $(dblock s' p)$
    by(auto simp add: HInv^4a1-def Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv^4a2:
[ HPhase1or2ReadThen $s s' p d r ; \ HInv^4a2 s q \implies HInv^4a2 s' q $]
by(auto simp add: Phase1or2ReadThen-def HInv^4a2-def)
lemma HPhase1or2ReadThen-HInv4a-p:
assumes act: HPhase1or2ReadThen s s' p d r
and inv: HInv4a s p
and inv2b: Inv2b s
shows HInv4a s'
proof
from act inv2b
have phase s p ∈ {1, 2}
  by (auto simp add: Phase1or2ReadThen-def Inv2b-def Inv2b-inner-def)
with act inv
show ?thesis
  by (auto simp add: HPhase1or2ReadThen-HInv4a1-p HPhase1or2ReadThen-HInv4a2)
qed

lemma HPhase1or2ReadThen-HInv4a-q:
assumes act: HPhase1or2ReadThen s s' p d r
and inv: HInv4a s q
and pnq: p ≠ q
shows HInv4a s'
proof
from act pnq
have blocksOf s' q ⊆ blocksOf s q
  by (auto simp add: Phase1or2ReadThen-def InitializePhase-def blocksOf-def rdBy-def)
with act inv pnq
show ?thesis
  by (auto simp add: Phase1or2ReadThen-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

theorem HPhase1or2ReadThen-HInv4a:
[ HPhase1or2ReadThen s s' p d r; HInv4a s q; Inv2b s ] ⇒ HInv4a s' q
by (blast dest: HPhase1or2ReadThen-HInv4a-p HPhase1or2ReadThen-HInv4a-q)

theorem HPhase1or2ReadElse-HInv4a:
assumes act: HPhase1or2ReadElse s s' p d r
and inv: HInv4a s q and inv2a: Inv2a s'
shows HInv4a s'
proof
from act have HStartBallot s s' p
  by (simp add: Phase1or2ReadElse-def)
with inv inv2a show ?thesis
  by (blast dest: HStartBallot-HInv4a )
qed

lemma HEndPhase1-HInv4a1:
assumes act: HEndPhase1 s s' p
and inv: HInv4a1 s p

60
shows $H_{Inv4a1}' s'$

proof(auto simp add: $H_{Inv4a1}$-def)

fix bk

assume bk: $bk \in \text{blocksOf} s'$

from bk $H_{EndPhase1}$-blocksOf[OF act]

have bk $\in \{\text{dblock} s' p\} \cup \text{blocksOf} s p$

by blast

with act inv

show $\text{bal} bk \leq \text{mbal} (\text{dblock} s' p)$

by(auto simp add: $H_{Inv4a}$-def $H_{Inv4a1}$-def $EndPhase1$-def)

qed

lemma $H_{EndPhase1}$-$H_{Inv4a2}$:

assumes act: $H_{EndPhase1} s s'$

and inv: $H_{Inv4a2} s p$

and inv2a: $Inv2a s$

shows $H_{Inv4a2} s' p$

proof(auto simp add: $H_{Inv4a2}$-def)

fix D

assume Dmaj: $D \in \text{MajoritySet}$

from inv Dmaj

have $\exists d \in D$. $\text{mbal} (\text{disk} s d p) \leq \text{mbal} (\text{dblock} s p)$

$\text{bal} (\text{disk} s d p) \leq \text{bal} (\text{dblock} s p)$

by(auto simp add: $H_{Inv4a2}$-def)

then obtain d

where d-cond: $d \in D$

$\text{mbal} (\text{disk} s d p) \leq \text{mbal} (\text{dblock} s p)$

$\text{bal} (\text{disk} s d p) \leq \text{bal} (\text{dblock} s p)$

by auto

have disk s d p $\in \text{blocksOf} s p$

by(auto simp add: blocksOf-def)

with inv2a

have $\text{bal} (\text{disk} s d p) \leq \text{mbal} (\text{disk} s d p)$

by(auto simp add: $Inv2a$-def $Inv2a$-inner-def $Inv2a$-innermost-def)

with act d-cond

have $d \in D$

$\text{mbal} (\text{disk} s' d p) \leq \text{mbal} (\text{dblock} s' p)$

$\text{bal} (\text{disk} s' d p) \leq \text{bal} (\text{dblock} s' p)$

by(auto simp add: $EndPhase1$-def)

with Dmaj

show $\exists d \in D$. $\text{mbal} (\text{disk} s' d p) \leq \text{mbal} (\text{dblock} s' p)$

$\text{bal} (\text{disk} s' d p) \leq \text{bal} (\text{dblock} s' p)$

by auto

qed

lemma $H_{EndPhase1}$-$H_{Inv4a-p}$:

assumes act: $H_{EndPhase1} s s'$

and inv: $H_{Inv4a} s p$

and inv2a: $Inv2a s$

61
shows $HInv4a \ s' \ p$

proof –

from act
have phase: $0 < \text{phase} \ s \ p$
  by (auto simp add: EndPhase1-def)
with act inv inv2a
show ?thesis
  by (auto simp del: HEndPhase1-def simp add: HInv4a-def)

qed

lemma HEndPhase1-HInv4a-q:
assumes act: $HEndPhase1 \ s \ s' \ p$
and inv: $HInv4a \ s \ q$
and pnq: $p \neq q$
shows $HInv4a \ s' \ q$

proof –

from act pnq
have dblock $s' \ q = \text{dblock} \ s \ q \land \text{disk} \ s' = \text{disk} \ s$
  by (auto simp add: EndPhase1-def)
moreover
from act pnq
have $\forall \ p \ d. \ \text{rdBy} \ s' \ q \ p \ d \subseteq \text{rdBy} \ s \ q \ p \ d$
  by (auto simp add: EndPhase1-def InitializePhase-def rdBy-def)

hence $(\text{UN} \ p \ d. \ \text{rdBy} \ s' \ q \ p \ d) \subseteq (\text{UN} \ p \ d. \ \text{rdBy} \ s \ q \ p \ d)$
  by (auto, blast)
ultimately
have blocksOf $s' \ q \subseteq \text{blocksOf} \ s \ q$
  by (auto simp add: blocksOf-def, blast)
with act inv pnq
show ?thesis
  by (auto simp add: EndPhase1-def HInv4a-def HInv4a1-def HInv4a2-def)

qed

theorem HEndPhase1-HInv4a-a:
[ $HEndPhase1 \ s \ s' \ p; \ HInv4a \ s \ q$ ] $\implies \ HInv4a \ s' \ q$
by (blast dest: HEndPhase1-HInv4a-a-p HEndPhase1-HInv4a-a-q)

theorem HFail-HInv4a:
[ $HFail \ s \ s' \ p; \ HInv4a \ s \ q$ ] $\implies \ HInv4a \ s' \ q$
by (auto simp add: Fail-def HInv4a-def HInv4a1-def HInv4a2-def InitializePhase-def
blocksOf-def rdBy-def)

theorem HPhase0Read-HInv4a:
[ $HPhase0Read \ s \ s' \ p \ d; \ HInv4a \ s \ q$ ] $\implies \ HInv4a \ s' \ q$
by (auto simp add: Phase0Read-def HInv4a-def HInv4a1-def HInv4a2-def InitializePhase-def
blocksOf-def rdBy-def)
theorem $HEndPhase2$-HINV4a:
\[
[ HEndPhase2 s s' p; HINV4a s q ] \implies HINV4a s' q
\]
by (auto simp add: EndPhase2-def HINV4a-def HINV4a1-def HINV4a2-def InitializePhase-def blocksOf-def rdBy-def)

lemma allSet:
assumes $aPQ$: $\forall a. \forall r \in P a. Q r$ and $rb$: $rb \in P d$
shows $Q rb$
proof
  from $aPQ$ have $\forall r \in P d. Q r$ by auto
  with $rb$
  show $?thesis$ by auto
qed

lemma $EndPhase0$-44:
assumes $act$: $EndPhase0 s s' p$
and $bk$: $bk \in \text{blocksOf } s p$
and $inv4d$: $HINV4d s p$
and $inv2c$: $Inv2c-inner s p$
shows $\exists d. \exists rb \in \text{blocksRead } s p d. \text{bal } bk \leq \text{mbal } (\text{block } rb)$
proof
  from $bk$ $inv4d$
  have $\exists D1 \in \text{MajoritySet}. \forall d \in D1. \text{bal } bk \leq \text{mbal } (\text{disk } s d p)$ — 4.2
    by (auto simp add: HINV4d-def)
  with $majorities-intersect$
  have $p43$: $\forall D \in \text{MajoritySet}. \exists d \in D. \text{bal } bk \leq \text{mbal } (\text{disk } s d p)$
    by (simp add: MajoritySet-def, blast)
  from $act$
  have $\forall d. \forall rb \in \text{blocksRead } s p d. \text{block } rb = \text{disk } s d p$ — 5.1
    by (simp add: Inv2c-inner-def)
  hence $\forall d. \text{hasRead } s p d$
    $\implies (\exists rb \in \text{blocksRead } s p d. \text{block } rb = \text{disk } s d p)$ — 5.2
    (is $\forall d. ?H d \implies ?P d$)
    by (auto simp add: hasRead-def)
  with $act$
  have $p53$: $\exists D \in \text{MajoritySet}. \forall d \in D. ?P d$
    by (auto simp add: MajoritySet-def $EndPhase0$-def)
  show $?thesis$
proof
  from $p43$ $p53$
  have $\exists D \in \text{MajoritySet}. (\exists d \in D. \text{bal } bk \leq \text{mbal } (\text{disk } s d p))$
    $\wedge (\forall d \in D. ?P d)$
    by $auto$
  thus $?thesis$
    by $force$
qed
proof (auto simp add: EndPhase0-def)
fix bk
assume bk ∈ blocksOf s' p
with HEndPhase0-blocksOf[OF act]
have bk ∈ {dblock s' p} ∪ blocksOf s p by auto
thus bal bk ≤ mbal (dblock s' p)
proof
assumption
have Inv2a-innermost s' p bk
by(auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
with bk show \ ?thesis
by(auto simp add: Inv2a-innermost-def)
next
assume bk: bk ∈ blocksOf s p
from act
have f1: \forall r ∈ allBlocksRead s p. mbal r < mbal (dblock s' p)
by(auto simp add: EndPhase0-def)
with act inv4d inv2c bk
have \exists d. \exists rb ∈ blocksRead s p d. bal bk ≤ mbal (block rb)
by(auto dest: EndPhase0-44)
with f1
show \ ?thesis
by(auto simp add: EndPhase0-def allBlocksRead-def allRdBls-def dest: allSet)
qed

lemma hasRead-allBlks:
assumes inv2c: Inv2c-inner s p
and phase: phase s p = 0
shows (\forall d ∈ {d. hasRead s p d p}. disk s d p ∈ allBlocksRead s p)
proof
fix d
assume d: d ∈ {d. hasRead s p d p} (is d ∈ ?D)
hence br-ne: blocksRead s p d ≠ {} by (auto simp add: hasRead-def)
from inv2c phase
have \forall br ∈ blocksRead s p d. block br = disk s d p
by(auto simp add: Inv2c-inner-def)
with br-ne
have disk s d p ∈ block ' blocksRead s p d
  by force
thus disk s d p ∈ allBlocksRead s p
  by(auto simp add: allBlocksRead-def allRdBlks-def)
qed

lemma HEndPhase0-41:
  assumes act: HEndPhase0 s s' p
  and inv1: Inv1 s
  and inv2c: Inv2c-inner s p
  shows ∃ D∈MajoritySet. ∀ d∈D. mbal(disk s d p) ≤ mbal(dblock s' p)
    ∧ bal(disk s d p) ≤ bal(dblock s' p)
proof –
  from act HEndPhase0-some[OF act inv1]
  have p51: ∀ br∈allBlocksRead s p. mbal br < mbal(dblock s' p)
    ∧ bal br ≤ bal(dblock s' p)
    and a: IsMajority({d. hasRead s p d p})
    and phase: phase s p = 0
    by(auto simp add: EndPhase0-def)+
  from inv2c phase
  have (∀ d∈{d. hasRead s p d p}. disk s d p ∈ allBlocksRead s p)
    by(auto dest: hasRead-allBlks)
  with p51
  have (∀ d∈{d. hasRead s p d p}. mbal(disk s d p) ≤ mbal(dblock s' p)
    ∧ bal(disk s d p) ≤ bal(dblock s' p))
    by force
  with a show ?thesis
    by(auto simp add: MajoritySet-def)
qed

lemma Majority-exQ:
  assumes asm1: ∃ D∈MajoritySet. ∀ d∈D. P d
  shows ∀ D∈MajoritySet. ∃ d∈D. P d
using asm1
proof(auto simp add: MajoritySet-def)
  fix D1 D2
  assume D1: IsMajority D1 and D2: IsMajority D2
  and Px: ∀ x∈D1. P x
  from D1 D2 majorities-intersect
  have ∃ d∈D1. d∈D2 by auto
  with Px
  show ∃ x∈D2. P x
    by auto
qed

lemma HEndPhase0-HInv4a2-p:
  assumes act: HEndPhase0 s s' p
  and inv1: Inv1 s
and \( \text{inv}2c: \text{Inv2c-inner} \ s \ p \)
shows \( H\text{Inv}4a2 \ s' \ p \)

proof \((\text{simp add: } H\text{Inv}4a2\text{-def})\)
from act
have \( \text{disk'}: \text{disk} \ s = \text{disk} \ s \)
by \((\text{simp add: EndPhase0-def})\)
from act \(\text{inv}2c\)
have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ m\text{bal}(\text{disk} \ s \ d \ p) \leq m\text{bal}(\text{dblock} \ s' \ p) \)
\& \( \text{bal}(\text{disk} \ s \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \)
by \((\text{blast dest: HEndPhase0-41})\)
from Majority-exQ \[\text{OF this}\]
have \( \forall D \in \text{MajoritySet}. \exists d \in D. \ m\text{bal}(\text{disk} \ s \ d \ p) \leq m\text{bal}(\text{dblock} \ s' \ p) \)
\& \( \text{bal}(\text{disk} \ s \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \)
(is \(?P (\text{disk} \ s)\) ) .
from ss subst \[\text{OF disk', of } ?P, \text{ OF this}\]
show \( \forall D \in \text{MajoritySet}. \exists d \in D. \ m\text{bal}(\text{disk} \ s' \ d \ p) \leq m\text{bal}(\text{dblock} \ s' \ p) \)
\& \( \text{bal}(\text{disk} \ s' \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \).
qed

lemma \( H\text{EndPhase0-HInv}4a-p:\)
assumes \( \text{act}: H\text{EndPhase0} \ s \ s' \ p \)
and \( \text{inv}2a: \text{Inv2a} \ s \)
and \( \text{inv}2: \text{Inv2c} \ s \)
and \( \text{inv}4d: H\text{Inv}4d \ s \ p \)
and \( \text{inv}1: \text{Inv1} \ s \)
and \( \text{inv}: H\text{Inv}4a \ s \ p \)
shows \( H\text{Inv}4a \ s' \ p \)
proof --
from \(\text{inv}2\)
have \(\text{inv}2c: \text{Inv2c-inner} \ s \ p \)
by \((\text{auto simp add: Inv2c-def})\)
with \(\text{inv}1 \ \text{inv}2a \ \text{act}\)
have \(\text{inv}2a': \text{Inv2a} \ s' \)
by \((\text{blast dest: HEndPhase0-Inv2a})\)
from act
have \(\text{phase} \ s' \ p = 1 \)
by \((\text{auto simp add: EndPhase0-def})\)
with \(\text{act} \ \text{inv} \ \text{inv}2c \ \text{inv}4d \ \text{inv}2a' \ \text{inv}1\)
show \(?\text{thesis}\)
by \((\text{auto simp add: HInv4a-def simp del: HEndPhase0-def elim: HEndPhase0-HInv4a1-p HEndPhase0-HInv4a2-p})\)
qed

lemma \( H\text{EndPhase0-HInv}4a-q:\)
assumes \( \text{act}: H\text{EndPhase0} \ s \ s' \ p \)
and \( \text{inv}: H\text{Inv}4a \ s \ q \)
and \(\text{pnq: } p \neq q\)
shows \( H\text{Inv}4a \ s' \ q \)
proof --
from Act pnq
have dblock s’ q = dblock s q ∧ disk s’ = disk s
  by (auto simp add: EndPhase0-def)
moreover
from Act pnq
have ∀p d. rdBy s’ q p d ⊆ rdBy s q p d
  by (auto simp add: EndPhase0-def InitializePhase-def rdBy-def)
hence (UN p d. rdBy s’ q p d) ⊆ (UN p d. rdBy s q p d)
  by (auto, blast)
ultimately
have blocksOf s’ q ⊆ blocksOf s q
  by (auto simp add: blocksOf-def, blast)
with Act inv pnq
show ?thesis
  by (auto simp add: EndPhase0-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

theorem HEndPhase0-HInv4a:
[ [ HEndPhase0 s s’ p; HInv4a s q; HInv4d s p; Inv2a s; Inv1 s; Inv2a s; Inv2c s ] ]
⇒ HInv4a s’ q
by (blast dest: HEndPhase0-HInv4a-p HEndPhase0-HInv4a-q)

C.4.2 Proofs of Invariant 4b

lemma blocksRead-allBlocksRead:
rb ∈ blocksRead s p d ⇒ block rb ∈ allBlocksRead s p
by (auto simp add: allBlocksRead-def allRdBlks-def)

lemma HEndPhase0-dblock-mbal:
[ [ HEndPhase0 s s’ p ] ]
⇒ ∀ br ∈ allBlocksRead s p. mbal br < mbal(dblock s’ p)
by (auto simp add: EndPhase0-def)

lemma HEndPhase0-HInv4b-p-dblock:
assumes act: HEndPhase0 s s’ p
and inv1: Inv1 s
and inv2a: Inv2a s
and inv2c: Inv2c-inner s p
shows bal(dblock s’ p) < mbal(dblock s’ p)
proof –
  from Act have phase s p = 0 by (auto simp add: EndPhase0-def)
  with inv2c
  have ∀ d. ∃ br ∈ blocksRead s p d. proc br = p ∧ block br = disk s d p
    by (auto simp add: Inv2c-inner-def)
hence allBlks-in-blocksOf: allBlocksRead s p ⊆ blocksOf s p
    by (auto simp add: allBlocksRead-def allRdBlks-def blocksOf-def)
  from Act HEndPhase0-some[OF act inv1]
have p53: \( \exists \, \text{br} \in \text{allBlocksRead s p} \). \( \text{bal}(\text{dblock s' p}) = \text{bal br} \)
  by (auto simp add: EndPhase0-def)
from inv2a
have i2: \( \forall \, p. \forall \, \text{bk} \in \text{blocksOf s p} \). \( \text{bal} \, \text{bk} \leq \text{mbal} \, \text{bk} \)
  by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def)
with allBlks-in-blocksOf
have \( \forall \, \text{bk} \in \text{allBlocksRead s p} \). \( \text{bal} \, \text{bk} \leq \text{mbal} \, \text{bk} \)
  by auto
with p53
have \( \exists \, \text{br} \in \text{allBlocksRead s p} \). \( \text{bal} \, (\text{dblock s' p}) \leq \text{mbal} \, \text{br} \)
  by force
with HEndPhase0-dblock-mbal[OF act]
show ?thesis
  by auto
qed

lemma HEndPhase0-HInv4b-p-blocksOf:
  assumes act: HEndPhase0 s s' p
  and inv4d: HInv4d s p
  and inv2c: Inv2c-inner s p
  and bk: \( \text{bk} \in \text{blocksOf s p} \)
  shows \( \text{bal} \, \text{bk} < \text{mbal}(\text{dblock s' p}) \)
proof –
  from inv4d majorities-intersect bk
have p43: \( \forall \, D \in \text{MajoritySet}. \exists \, d \in D. \, \text{bal} \, \text{bk} \leq \text{mbal}(\text{disk s d p}) \)
    by (auto simp add: HInv4d-def MajoritySet-def Majority-exQ)
have \( \exists \, \text{br} \in \text{allBlocksRead s p} \). \( \text{bal} \, \text{bk} \leq \text{mbal} \, \text{br} \)
  proof –
    from act
    have maj: \( \text{IsMajority}\{d. \, \text{hasRead s p d p}\}\) (is IsMajority(?D))
      and phase: \( \text{phase} \, \text{s p} = 0 \)
      by (simp add: EndPhase0-def)+
    have br-ne: \( \forall \, d \in ?D. \, \text{blocksRead s p d} \neq \{\} \)
      by (auto simp add: hasRead-def)
    from phase inv2c
    have \( \forall \, d \in ?D. \forall \, \text{br} \in \text{blocksRead s d p} \). \( \text{block} \, \text{br} = \text{disk} \, \text{s d p} \)
      by (auto simp add: Inv2c-inner-def)
    with br-ne
    have \( \forall \, d \in ?D. \exists \, \text{br} \in \text{allBlocksRead s p} \, \text{br} = \text{disk} \, \text{s d p} \)
      by (blast dest: blocksRead-allBlocksRead)
    with p43 maj
    show ?thesis
      by (auto simp add: MajoritySet-def)
  qed
with HEndPhase0-dblock-mbal[OF act]
show ?thesis
  by auto
qed
lemma HEndPhase0-HInv4b-p:
assumes act: HEndPhase0 s s' p
and inv4d: HInv4d s p
and inv1: Inv1 s
and inv2a: Inv2a s
and inv2c: Inv2c-inner s p
shows HInv4b s' p
proof(clarsimp simp add: HInv4b-def)
from act
have phase: phase s p = 0
  by(auto simp add: EndPhase0-def)
fix bk
assume bk: bk∈ blocksOf s' p
with HEndPhase0-blocksOf[OF act]
have bk∈{dblock s' p} ∨ bk∈blocksOf s p
  by blast
thus bal bk < mbal (dblock s' p)
proof
  assume bk: bk∈{dblock s' p}
  with act inv1 inv2a inv2c
  show ?thesis
    by(auto simp del: HEndPhase0-def
dest: HEndPhase0-HInv4b-p-dblock )
next
  assume bk: bk ∈ blocksOf s p
  with act inv2c inv4d
  show ?thesis
    by(blast dest: HEndPhase0-HInv4b-p-blocksOf)
qed
qed

lemma HEndPhase0-HInv4b-q:
assumes act: HEndPhase0 s s' p
and pnq: p≠q
and inv: HInv4b s q
shows HInv4b s' q
proof
  from act pnq
  have disk': disk s'='disk s
    and dblock': dblock s' q='dblock s q
    and phase': phase s' q='phase s q
    by(auto simp add: EndPhase0-def)
  from act pnq
  have blocksRead': ∀ q. allRdBlks s' q ⊆ allRdBlks s q
    by(auto simp add: EndPhase0-def InitializePhase-def allRdBlks-def)
with disk' dblock'
  have blocksOf s' q ⊆ blocksOf s q
    by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis
  by (auto simp add: HInv4b-def)
qed

theorem HEndPhase0-HInv4b:
  assumes act: HEndPhase0 s s' p
  and inv4b: HInv4b s q
  and inv4d: HInv4d s p
  and inv1: Inv1 s
  and inv2a: Inv2a s
  and inv2c: Inv2c s p
  shows HInv4b s' q
proof (cases p = q)
  case True
  with HEndPhase0-HInv4b-p[OF act inv4d inv1 inv2a inv2c]
  show ?thesis by simp
next
  case False
  from HEndPhase0-HInv4b-q[OF act False inv]
  show ?thesis.
qed

lemma HStartBallot-HInv4b-p:
  assumes act: HStartBallot s s' p
  and inv2a: Inv2a-innermost s p (dblock s p)
  and inv4b: HInv4b s p
  and inv4a: HInv4a s p
  shows HInv4b s' p
proof (clarsimp simp add: HInv4b-def)
  fix bk
  assume bk: bk ∈ blocksOf s' p
  from act
  have phase': phase s' p = 1
    and phase: phase s p ∈ {1,2}
    by (auto simp add: StartBallot-def)
  from act
  have p42: mbal (dblock s p) < mbal (dblock s' p)
    ∧ bal(dblock s p) = bal(dblock s' p)
    by (auto simp add: StartBallot-def)
  from HStartBallot-blocksOf[OF act] bk
  have bk ∈ {dblock s' p} ∪ blocksOf s p
    by blast
  thus bal bk < mbal (dblock s' p)
proof
  assume bk: bk ∈ {dblock s' p}
  from inv2a
  have bal (dblock s p) ≤ mbal (dblock s p)
    by (auto simp add: Inv2a-innermost-def)
  with p42 bk
show \( ? \text{thesis} \) by auto

next

assume \( bk: bk \in \text{blocksOf } s \ p \)
from phase \( \text{inv4}a \)
have \( p41: H\text{inv4}a1 \ s \ p \)
  by (auto simp add: \( H\text{inv4}a\)-def)
with \( p42 \ bk \)
show \( ? \text{thesis} \)
  by (auto simp add: \( H\text{inv4}a1\)-def)
qed

lemma \( H\text{StartBallot-}H\text{inv4}b\)-q:
  assumes act: \( H\text{StartBallot } s \ s' \ p \)
  and png: \( p \neq q \)
  and inv: \( H\text{inv4}b \ s \ q \)
  shows \( H\text{inv4}b \ s' \ q \)
proof -
  from act png
  have disk': \( \text{disk } s' = \text{disk } s \)
    and dblock': \( \text{dblock } s' \ q = \text{dblock } s \ q \)
    and phase': \( \text{phase } s' \ q = \text{phase } s \ q \)
      by (auto simp add: \( \text{StartBallot-def} \))
  from act png
  have blocksRead': \( \forall q. \text{allRdBlks } s' \ q \subseteq \text{allRdBlks } s \ q \)
    by (auto simp add: \( \text{StartBallot-def InitializePhase-def allRdBlks-def} \))
  with \( \text{disk'} \ \text{dblock'} \)
  have blocksOf s' q \( \subseteq \) blocksOf s q
    by (auto simp add: \( \text{blocksOf-def rdBy-def , blast} \))
  with inv phase' dblock'
  show \( ? \text{thesis} \)
    by (auto simp add: \( H\text{inv4}b\)-def)
qed

theorem \( H\text{StartBallot-}H\text{inv4}b\):
  assumes act: \( H\text{StartBallot } s \ s' \ p \)
  and inv2a: \( \text{inv2a } s \)
  and inv4b: \( H\text{inv4}b \ s \ q \)
  and inv4a: \( H\text{inv4}a \ s \ p \)
  shows \( H\text{inv4}b \ s' \ q \)
using act inv2a inv4b inv4a
proof (cases \( p = q \))
case True
  from inv2a
  have inv2a-innermost s p (dblock s p)
    by (auto simp add: \( \text{inv2a-def inv2a-inner-def blocksOf-def} \))
  with act True inv4b inv4a
  show \( ? \text{thesis} \)
    by (blast dest: \( H\text{StartBallot-}H\text{inv4}b\)-p)

71
next
case False
with act inv4b
show ?thesis
  by(blast dest: HStartBallot-HInv4b-q)
qed

theorem HPhase1or2Write-HInv4b:
  [ HPhase1or2Write s s' p d; HInv4b s q ] \implies HInv4b s' q
by(auto simp add: Phase1or2Write-def HInv4b-def
  blocksOf-def rdBy-def)

lemma HPhase1or2ReadThen-HInv4b-p:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv4b s p
  shows HInv4b s' p
proof
  from HPhase1or2ReadThen-blocksOf[OF act] inv act
  show ?thesis
    by(auto simp add: HInv4b-def Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv4b-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4b s q
  and pnq: p \neq q
  shows HInv4b s' q
using assms HPhase1or2ReadThen-blocksOf[OF act]
by(auto simp add: Phase1or2ReadThen-def HInv4b-def)

theorem HPhase1or2ReadThen-HInv4b:
  [ HPhase1or2ReadThen s s' p d q; HInv4b s r ] \implies HInv4b s' r
by(blast dest: HPhase1or2ReadThen-HInv4b-p
  HPhase1or2ReadThen-HInv4b-q)

theorem HPhase1or2ReadElse-HInv4b:
  [ HPhase1or2ReadElse s s' p d q; HInv4b s r;
    Inv2a s; HInv4a s p ]
  \implies HInv4b s' r
using HStartBallot-HInv4b
by(auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase1-HInv4b-p:
  HEndPhase1 s s' p \implies HInv4b s' p
by(auto simp add: EndPhase1-def HInv4b-def)

lemma HEndPhase1-HInv4b-q:
  assumes act: HEndPhase1 s s' p
  and pnq: p \neq q

72
and \( \text{inv: } \text{HInv4b s q} \)
shows \( \text{HInv4b s' q} \)

proof (cases \( p=q \))
case True
with \( \text{HEndPhase1-HInv4b-p[OF act]} \)
show \( \text{?thesis by simp} \)
next
case False
from \( \text{HEndPhase1-HInv4b-q[OF act False inv]} \)
show \( \text{?thesis} \).

qed

theorem \( \text{HEndPhase1-HInv4b:} \)
assumes \( \text{act: } \text{HEndPhase1 s s' p} \)
and \( \text{inv: } \text{HInv4b s q} \)
shows \( \text{HInv4b s' q} \)

proof

lemma \( \text{HEndPhase2-HInv4b-p:} \)
\( \text{HEndPhase2 s s' p } \implies \text{HInv4b s' p} \)
by (auto simp add: \( \text{EndPhase2-def HInv4b-def} \))

lemma \( \text{HEndPhase2-HInv4b-q:} \)
assumes \( \text{act: } \text{HEndPhase2 s s' p} \)
and \( \text{pq: } p\neq q \)
and \( \text{inv: } \text{HInv4b s q} \)
shows \( \text{HInv4b s' q} \)

proof (cases \( p=q \))

have \( \text{disk': disk s'=disk s} \)
and \( \text{dblock': dblock s' q=dblock s q} \)
and \( \text{phase': phase s' q =phase s q} \)
by (auto simp add: \( \text{EndPhase2-def} \))

from act pq
have \( \text{blocksRead': } \forall q. \text{ allRdBlks s' q } \subseteq \text{ allRdBlks s q} \)
by (auto simp add: \( \text{EndPhase1-def} \text{ InitializePhase-def allRdBlks-def} \))

with \( \text{disk' dblock'} \)
have \( \text{blocksOf s' q } \subseteq \text{ blocksOf s q} \)
by (auto simp add: \( \text{allRdBlks-def blocksOf-def rdBy-def blast} \))

with \( \text{inv phase' dblock'} \)
show \( \text{?thesis} \)
by (auto simp add: \( \text{HInv4b-def} \))

qed
have blocksRead': \forall q. allRdBlks s' q \subseteq allRdBlks s q 
by(auto simp add: EndPhase2-def InitializePhase-def allRdBlks-def)
with disk' dblock'

have blocksOf s' q \subseteq blocksOf s q 
by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis 
by(auto simp add: HInv4b-def)
qed

theorem HEndPhase2-HInv4b:
assumes act: HEndPhase2 s s' p
and inv: HInv4b s q
shows HInv4b s' q
proof(cases p=q)
case True
with HEndPhase2-HInv4b-p[OF act]
show ?thesis by simp
next
case False
from HEndPhase2-HInv4b-q[OF act False inv]
show ?thesis .
qed

lemma HFail-HInv4b-p:
HFail s s' p \Rightarrow HInv4b s' p 
by(auto simp add: Fail-def HInv4b-def)

lemma HFail-HInv4b-q:
assumes act: HFail s s' p 
and pnq: p\neq q
and inv: HInv4b s q
shows HInv4b s' q
proof —
from act pnq
have disk': disk s'=disk s
and dblock': dblock s' q= dblock s q
and phase': phase s' q = phase s q 
by(auto simp add: Fail-def)
from act pnq
have blocksRead': \forall q. allRdBlks s' q \subseteq allRdBlks s q 
by(auto simp add: Fail-def InitializePhase-def allRdBlks-def)
with disk' dblock'
have blocksOf s' q \subseteq blocksOf s q 
by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis 
by(auto simp add: HInv4b-def)
qed
theorem HFail-HInv4b:
  assumes act: HFail s s' p
  and inv: HInv4b s q
  shows HInv4b s' q
proof (cases p=q)
  case True
    with HFail-HInv4b-p[OF act]
    show ?thesis by simp
next
  case False
    from HFail-HInv4b-q[OF act False inv]
    show ?thesis .
qed

lemma HPhase0Read-HInv4b-p:
  HPhase0Read s s' p d ⇒ HInv4b s' p
by (auto simp add: Phase0Read-def HInv4b-def)

lemma HPhase0Read-HInv4b-q:
  assumes act: HPhase0Read s s' p d
  and pq: p≠q
  and inv: HInv4b s q
  shows HInv4b s' q
proof (cases p=q)
  from act pq
  have disk': disk s'=disk s
  and dblock': dblock s' q=dblock s q
  and phase': phase s' q =phase s q
  by (auto simp add: Phase0Read-def)
  from HPhase0Read-blocksOf[OF act] inv phase' dblock'
  show ?thesis
  by (auto simp add: HInv4b-def)
qed

theorem HPhase0Read-HInv4b:
  assumes act: HPhase0Read s s' p d
  and inv: HInv4b s q
  shows HInv4b s' q
proof (cases p=q)
  case True
    with HPhase0Read-HInv4b-p[OF act]
    show ?thesis by simp
next
  case False
    from HPhase0Read-HInv4b-q[OF act False inv]
    show ?thesis .
qed
C.4.3 Proofs of Invariant 4c

lemma \textit{HStartBallot-HInv4c-p}:
\begin{align*}
\text{\tt[}\ [ \text{HStartBallot} \ s \ s' \ p; \ HInv4c \ s \ p \text{]} \implies HInv4c \ s' \ p \text{]} \end{align*}
by (auto simp add: \texttt{StartBallot-def \ HInv4c-def})

lemma \textit{HStartBallot-HInv4c-q}:
\begin{align*}
\text{assumes act: } \text{HStartBallot} \ s \ s' \ p \\
\text{and inv: } \text{HInv4c} \ s \ q \\
\text{and } \texttt{\textit{pnq} \ p \neq q} \\
\text{shows } \text{HInv4c} \ s' \ q
\end{align*}
proof -
from \texttt{act \ \texttt{\textit{pnq}}}
have \texttt{phase: phase} \ s' \ q = \texttt{phase} \ s \ q \\
and \texttt{dblock: dblock} \ s \ q = \texttt{dblock} \ s' \ q \\
and \texttt{disk: disk} \ s' \ = \texttt{disk} \ s \\
by (auto simp add: StartBallot-def)
with \texttt{\textit{inv}}
show \texttt{?thesis}
by (auto simp add: HInv4c-def)
qed

theorem \textit{HStartBallot-HInv4c}:
\begin{align*}
\text{\tt[}\ [ \text{HStartBallot} \ s \ s' \ p; \ HInv4c \ s \ q \text{]} \implies HInv4c \ s' \ q \end{align*}
by (blast dest: \texttt{HStartBallot-HInv4c-p \ HStartBallot-HInv4c-q})

lemma \textit{HPhase1or2Write-HInv4c-p}:
\begin{align*}
\text{assumes act: } \text{HPhase1or2Write} \ s \ s' \ p \ d \\
\text{and inv: } \text{HInv4c} \ s \ p \\
\text{and inv2c: } \text{Inv2c} \ s \\
\text{shows } \text{HInv4c} \ s' \ p
\end{align*}
proof (cases phase \ s' \ p = 2)
assume \texttt{phase': phase} \ s' \ p = 2
show \texttt{?thesis}
proof (auto simp add: HInv4c-def phase' MajoritySet-def)
from \texttt{\textit{act \ phase'}}
have bal: \texttt{bal(\texttt{dblock} \ s' \ p)=bal(\texttt{dblock} \ s \ p)} \\
and \texttt{phase: phase} \ s \ p = 2 \\
by (auto simp add: Phase1or2Write-def)
from \texttt{\textit{phase'} inv2c \ act}
have mbal(\texttt{disk} \ s' \ d \ p)=\texttt{bal(\texttt{dblock} \ s \ p)} \\
by (auto simp add: Phase1or2Write-def Inv2c-def \ Inv2c-inner-def)
with bal
have bal(\texttt{dblock} \ s' \ p) = mbal(\texttt{disk} \ s' \ d \ p) \\
by auto
with \texttt{\textit{inv \ phase \ act}}
show \exists D. \texttt{. \texttt{IsMajority} \ D} \\
\wedge (\forall \texttt{d} \in \texttt{D}. \texttt{mbal(\texttt{disk} \ s' \ d \ p)} = \texttt{bal(\texttt{dblock} \ s' \ p)}) \\
by (auto simp add: HInv4c-def Phase1or2Write-def MajoritySet-def)
qed

76
next
case False
with act
show ?thesis
  by (auto simp add: HInv4c-def Phase1or2Write-def)
qed

lemma HPhase1or2Write-HInv4c-q:
  assumes act: HPhase1or2Write s s' p d
  and inv: HInv4c s q
  and pnq: p ≠ q
  shows HInv4c s' q
proof –
  from act pnq
  have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: ∀ d. disk s' d q = disk s d q
  by (auto simp add: Phase1or2Write-def)
  with inv
  show ?thesis
  by (auto simp add: HInv4c-def)
qed

theorem HPhase1or2Write-HInv4c:
  [ HPhase1or2Write s s' p d; HInv4c s q; Inv2c s ]
  ⇒ HInv4c s' q
by (blast dest: HPhase1or2Write-HInv4c-p
          HPhase1or2Write-HInv4c-q)

lemma HPhase1or2ReadThen-HInv4c-p:
  [ HPhase1or2ReadThen s s' p d q; HInv4c s p ]
  ⇒ HInv4c s' p
by (auto simp add: Phase1or2ReadThen-def HInv4c-def)

lemma HPhase1or2ReadThen-HInv4c-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4c s q
  and pnq: p ≠ q
  shows HInv4c s' q
proof –
  from act pnq
  have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: disk s' = disk s
  by (auto simp add: Phase1or2ReadThen-def)
  with inv
  show ?thesis
  by (auto simp add: HInv4c-def)
qed
theorem HPhase1or2ReadThen-HInv4c:
\[ \begin{array}{l}
[ \text{HPhase1or2ReadThen }\ s \ s' \ p \ d \ r; \ \text{HInv4c }\ s \ q] \\
\implies \text{HInv4c }\ s' \ q
\end{array} \]
by (blast dest: HPhase1or2ReadThen-HInv4c-p HPhase1or2ReadThen-HInv4c-q)

theorem HPhase1or2ReadElse-HInv4c:
\[ \begin{array}{l}
[ \text{HPhase1or2ReadElse }\ s \ s' \ p \ d \ r; \ \text{HInv4c }\ s \ q] \\
\implies \text{HInv4c }\ s' \ q
\end{array} \]
using HStartBallot-HInv4c
by (auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase1-HInv4c-p:
assumes act: HEndPhase1 s s' p
and inv2b: Inv2b s
shows HInv4c s' p
proof -
from act have maj: IsMajority \{d. d \in disksWritten s \ p \\
\wedge (\forall q \in (\text{UNIV} - \{p\}). \text{hasRead } s \ p \ d \ q)\}
(is IsMajority ?M)
by (simp add: EndPhase1-def)
from inv2b have \(\forall d \in ?M. \text{disk } s \ d \ p = \text{dblock } s \ p\)
by (auto simp add: Inv2b-def Inv2b-inner-def)
with act maj
show ?thesis
by (auto simp add: HInv4c-def EndPhase1-def MajoritySet-def EndPhase1-def)
qed

lemma HEndPhase1-HInv4c-q:
assumes act: HEndPhase1 s s' p
and inv: HInv4c s q
and pnq: p \neq q
shows HInv4c s' q
proof -
from act pnq have phase: phase s' q = phase s q
and dblock: dblock s q = dblock s' q
and disk: disk s' = disk s
by (auto simp add: EndPhase1-def)
with inv
show ?thesis
by (auto simp add: HInv4c-def)
qed

theorem HEndPhase1-HInv4c:
\[ \begin{array}{l}
[ \text{HEndPhase1 }\ s \ s' \ p; \ \text{HInv4c }\ s \ q; \ \text{Inv2b } s] \\
\implies \text{HInv4c }\ s' \ q
\end{array} \]
by (blast dest: HEndPhase1-HInv4c-p HEndPhase1-HInv4c-q)
lemma HEndPhase2-HInv4c-p:
[ HEndPhase2 s s' p ; HInv4c s p ] \Rightarrow HInv4c s' p
by (auto simp add: EndPhase2-def HInv4c-def)

lemma HEndPhase2-HInv4c-q:
assumes act : HEndPhase2 s s' p
and inv : HInv4c s q
and pnq : p \neq q
shows HInv4c s' q
proof -
from act pnq
have phase : phase s' q = phase s q
and dblock : dblock s q = dblock s' q
and disk : disk s' = disk s
by (auto simp add: EndPhase2-def)
with inv
show ?thesis
by (auto simp add: HInv4c-def)
qed

theorem HEndPhase2-HInv4c:
[ HEndPhase2 s s' p ; HInv4c s q ] \Rightarrow HInv4c s' q
by (blast dest: HEndPhase2-HInv4c-p HEndPhase2-HInv4c-q)

lemma HFail-HInv4c-p:
[ HFail s s' p ; HInv4c s p ] \Rightarrow HInv4c s' p
by (auto simp add: Fail-def HInv4c-def)

lemma HFail-HInv4c-q:
assumes act : HFail s s' p
and inv : HInv4c s q
and pnq : p \neq q
shows HInv4c s' q
proof -
from act pnq
have phase : phase s' q = phase s q
and dblock : dblock s q = dblock s' q
and disk : disk s' = disk s
by (auto simp add: Fail-def)
with inv
show ?thesis
by (auto simp add: HInv4c-def)
qed

theorem HFail-HInv4c:
[ HFail s s' p ; HInv4c s q ] \Rightarrow HInv4c s' q
by (blast dest: HFail-HInv4c-p HFail-HInv4c-q)

lemma HPhase0Read-HInv4c-p:
lemma HPhase0Read-HInv4c-q:
assumes act: HPhase0Read s s' p d
and inv: HInv4c s q
and pnq: p≠q
shows HInv4c s' q
proof –
from act pnq
have phase: phase s' q = phase s q
and dblock: dblock s q = dblock s' q
and disk: disk s' = disk s
by(auto simp add: Phase0Read-def)
with inv
show ?thesis
by(auto simp add: HInv4c-def)
qed

theorem HPhase0Read-HInv4c:
[ HPhase0Read s s' p d; HInv4c s p ] ⇒ HInv4c s' p
by(blast dest: HPhase0Read-HInv4c-p HPhase0Read-HInv4c-q)

lemma HEndPhase0-HInv4c-p:
[ HEndPhase0 s s' p; HInv4c s p ] ⇒ HInv4c s' p
by(auto simp add: EndPhase0-def HInv4c-def)

lemma HEndPhase0-HInv4c-q:
assumes act: HEndPhase0 s s' p
and inv: HInv4c s q
and pnq: p≠q
shows HInv4c s' q
proof –
from act pnq
have phase: phase s' q = phase s q
and dblock: dblock s q = dblock s' q
and disk: disk s' = disk s
by(auto simp add: EndPhase0-def)
with inv
show ?thesis
by(auto simp add: HInv4c-def)
qed

theorem HEndPhase0-HInv4c:
[ HEndPhase0 s s' p; HInv4c s q ] ⇒ HInv4c s' q
by(blast dest: HEndPhase0-HInv4c-p HEndPhase0-HInv4c-q)
C.4.4 Proofs of Invariant 4d

lemma HStartBallot-HInv4d-p:
  assumes act: HStartBallot s s' p
  and inv: HInv4d s p
  shows HInv4d s' p
proof(clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk ∈ blocksOf s' p
  from act have bal': bal (dblock s' p) = bal (dblock s p)
    by(auto simp add: StartBallot-def)
  from subsetD[OF HStartBallot-blocksOf[OF act] bk]
  have ∃ D∈MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s d p)
proof
  assume bk: bk ∈ blocksOf s p
  with inv show ?thesis
    by(auto simp add: HInv4d-def)
next
  assume bk: bk ∈ {dblock s' p}
  with bal' inv
  show ?thesis
    by(auto simp add: HInv4d-def blocksOf-def)
qed
  with act show ∃ D∈MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s' d p)
    by(auto simp add: StartBallot-def)
qed

lemma HStartBallot-HInv4d-q:
  assumes act: HStartBallot s s' p
  and inv: HInv4d s q
  and pnq: p ≠ q
  shows HInv4d s' q
proof
  from act pnq
  have disk': disk s' = disk s
    by(auto simp add: StartBallot-def)
  and dblock': dblock s' q = dblock s q
  from act pnq
  have blocksRead': ∀ q. allRdBlks s' q ⊆ allRdBlks s q
    by(auto simp add: InitializePhase-def allRdBlks-def)
  with disk' dblock'
  have blocksOf s' q ⊆ blocksOf s q
    by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  from subsetD[OF this] inv
  have ∀ bk∈blocksOf s' q.
    ∃ D∈MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s d q)
    by(auto simp add: HInv4d-def)
  qed
with disk' show ?thesis by (auto simp add: HInv4d-def) qed

theorem HStartBallot-HInv4d:
[ HStartBallot s s' p; HInv4d s q ] \implies HInv4d s' q
by (blast dest: HStartBallot-HInv4d-p HStartBallot-HInv4d-q)

lemma HPhase1or2Write-HInv4d-p:
assumes act: HPhase1or2Write s s' p d
and inv: HInv4d s p
and inv4a: HInv4a s p
shows HInv4d s' p
proof (clarsimp simp add: HInv4d-def)
fix bk
assume bk: bk \in blocksOf s' p
from act have ddisk: \( \forall d. \) disk s' dd p = (if d = dd
then dblock s p
else disk s dd p)
and phase: phase s p \neq 0
by (auto simp add: Phase1or2Write-def)
from inv subsetD[OF HPhase1or2Write-blocksOf[OF act] bk]
have asm3: \( \exists D\in MajoritySet. \forall dd \in D. \) bal bk \leq mbal (disk s dd p)
by (auto simp add: HInv4d-def)
from phase inv4a subsetD[OF HPhase1or2Write-blocksOf[OF act] bk] ddisk
have p41: bal bk \leq mbal (disk s' d p)
by (auto simp add: HInv4a-def HInv4a1-def)
with ddisk asm3
show \( \exists D\in MajoritySet. \forall dd \in D. \) bal bk \leq mbal (disk s' dd p)
by (auto simp add: MajoritySet-def split: split-if-asm)
qed

lemma HPhase1or2Write-HInv4d-q:
assumes act: HPhase1or2Write s s' p d
and inv: HInv4d s q
and pnq: p\#q
shows HInv4d s' q
proof
from act pnq
have disk': \( \forall d. \) disk s' d q = disk s d q
by (auto simp add: Phase1or2Write-def)
from act pnq
have blocksRead': \( \forall q. \) allRdBlks s' q \subseteq allRdBlks s q
by (auto simp add: Phase1or2Write-def InitializePhase-def allRdBlks-def)
with act pnq
have blocksOf s' q \subseteq blocksOf s q
by (auto simp add: Phase1or2Write-def allRdBlks-def blocksOf-def rdBy-def)
from subsetD \[OF this\] inv
have \( \forall bk \in \text{blocksOf } s' q . \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal} (\text{disk } s \ d \ q) \)
  by (auto simp add: HInv4d-def)
with disk'
show \( ?\text{thesis} \)
by (auto simp add: HInv4d-def)
qed

theorem HPhase1or2Write-HInv4d:
[ \ HPhase1or2Write s s' p d; HInv4d s q; HInv4a s p \] \( \Rightarrow \) HInv4d s' q
by (blast dest: HPhase1or2Write-HInv4d-p HPhase1or2Write-HInv4d-q)

lemma HPhase1or2ReadThen-HInv4d-p:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv4d s p
  shows HInv4d s' p
proof (clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk \( \in \) blocksOf s' p
  from act
  have bal': \( \text{bal} (\text{dblock } s' p) = \text{bal} (\text{dblock } s \ p) \)
    by (auto simp add: Phase1or2ReadThen-def)
  from subsetD \[OF HPhase1or2ReadThen-blocksOf [OF act] bk\] inv
  have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal} (\text{disk } s \ d \ p) \)
    by (auto simp add: HInv4d-def)
  with act
  show \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal} (\text{disk } s' d \ p) \)
    by (auto simp add: Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv4d-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4d s q
  and pnq: p \( \neq \) q
  shows HInv4d s' q
proof
  from act pnq
  have disk': \( \text{disk } s' = \text{disk } s \)
    by (auto simp add: Phase1or2ReadThen-def)
  from act pnq
  have blocksOf s' q \( \subseteq \) blocksOf s q
    by (auto simp add: Phase1or2ReadThen-def allRdBlks-def blocksOf-def rdBy-def)
  from subsetD \[OF this\] inv
  have \( \forall bk \in \text{blocksOf } s' q . \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal} (\text{disk } s \ d \ q) \)

83
by (auto simp add: \(HInv4d\)-def)
with disk'
show ?thesis
by (auto simp add: \(HInv4d\)-def)
qed

theorem \(HPhase1or2ReadThen-HInv4d\):
\[
\begin{array}{l}
\forall s, s', p, d, r. \quad HInv4d s q \rightarrow HInv4d s' q
\end{array}
\]
by (blast dest: \(HPhase1or2ReadThen-HInv4d\)-p
\(HPhase1or2ReadThen-HInv4d\)-q)

theorem \(HPhase1or2ReadElse-HInv4d\):
\[
\begin{array}{l}
\forall s, s', p, d, r. \quad HInv4d s q \rightarrow HInv4d s' q
\end{array}
\]
using \(HStartBallot-HInv4d\)
by (auto simp add: \(Phase1or2ReadElse\)-def)

lemma \(HEndPhase1-HInv4d\)-p:
assumes act: \(HEndPhase1\) s s' p
and inv: \(HInv4d\) s p
and inv2b: \(Inv2b\) s
and inv4c: \(HInv4c\) s p
shows \(HInv4d\) s' p
proof (clarsimp simp add: \(HInv4d\)-def)
fix bk
assume bk: \(bk \in \text{blocksOf } s' p\)
from \(HEndPhase1-HInv4c\)-p[OF act inv4c inv2b]
have \(HInv4c\) s' p .
with act
have \(p31: \exists D \in \text{MajoritySet}. \quad \forall d \in D. \quad \text{mbal}(\text{disk } s' d p) = \text{bal}(\text{dblock } s' d p)\)
and disk': disk s' = disk s
by (auto simp add: \(EndPhase1\)-def \(HInv4c\)-def)
from \(subsetD\)(\(\exists D \in \text{MajoritySet}. \quad \forall d \in D. \quad \text{bal } bk \leq \text{mbal}(\text{disk } s' d p)\)
show \(\exists D \in \text{MajoritySet}. \quad \forall d \in D. \quad \text{bal } bk \leq \text{mbal}(\text{disk } s' d p)\)
proof
assume bk: \(bk \in \text{blocksOf } s p\)
with inv disk'
show ?thesis
by (auto simp add: \(HInv4d\)-def)
next
assume bk: \(bk \in \{\text{dblock } s' p\}\)
with p31
show ?thesis
by force
qed

lemma \(HEndPhase1-HInv4d\)-q:
assumes act: \(HEndPhase1\) s s' p
and inv: $H_{inv4d}\ s\ q$
and pnq: $p\neq q$
shows $H_{inv4d}\ s'\ q$

proof

- from act pnq
  have disk': disk $s' = disk\ s$
    and dblock': dblock $s'\ q = dblock\ s\ q$
    by (auto simp add: EndPhase1-def)
  from act pnq
  have blocksRead': $\forall q.\ allRdBlks s\ s' q \subseteq allRdBlks\ s\ q$
    by (auto simp add: EndPhase1-def InitializePhase-def allRdBlks-def)
  with disk'
    have blocksOf $s' q \subseteq blocksOf\ s\ q$
      by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  from subsetD[OF this] inv
  have $\forall bk \in blocksOf s' q$.
    $\exists D \in MajoritySet.\ \forall d \in D.\ bal bk \leq mbal(d disk s\ q)$
    by (auto simp add: HInv4d-def)
  with disk'
  show ?thesis
    by (auto simp add: Hinv4d-def)
  qed

theorem HEndPhase1-HInv4d:

[ HEndPhase1 $s\ s'\ p;\ H_{inv4d}\ s\ q;\ Inv2b\ s;\ H_{inv4c}\ s\ p]\n  \Rightarrow H_{inv4d}\ s' q
by (blast dest: HEndPhase1-HInv4d-p HEndPhase1-HInv4d-q)

lemma HEndPhase2-Hinv4d-p:
assumes act: HEndPhase2 $s\ s'\ p$
and inv: $H_{inv4d}\ s\ p$
shows $H_{inv4d}\ s'\ p$
proof (clarsimp simp add: Hinv4d-def)
  fix bk
  assume bk: $bk \in blocksOf s' p$
  from act
  have bal': $bal\ (dblock\ s' p) = bal\ (dblock\ s\ p)$
    by (auto simp add: EndPhase2-def)
  from subsetD[OF HEndPhase2-blocksOf[OF act] bk] inv
  have $\exists D \in MajoritySet.\ \forall d \in D.\ bal bk \leq mbal\ (disk\ s\ d\ p)$
    by (auto simp add: Hinv4d-def)
  with act
  show $\exists D \in MajoritySet.\ \forall d \in D.\ bal bk \leq mbal\ (disk\ s'\ d\ p)$
    by (auto simp add: EndPhase2-def)
  qed

lemma HEndPhase2-Hinv4d-q:
assumes act: HEndPhase2 $s\ s'\ p$

85
and inv: $H_{inv4d} \ s \ q$
and $pnq: p \neq q$
shows $H_{inv4d} \ s' \ q$

proof –

from act $pnq$
have $disk': disk \ s' = disk \ s$
  by(auto simp add: EndPhase2-def)
from act $pnq$
have $blocksOf \ s' \ q \subseteq blocksOf \ s \ q$
  by(auto simp add: EndPhase2-def InitializePhase-def
      allRdBlks-def blocksOf-def rdBy-def)
from subsetD[OF this] inv
have $\forall bk \in blocksOf \ s' \ q$.  
  $\exists D \in MajoritySet. \ \forall d \in D. \ bal \ bk \leq mbal(disk \ s \ d \ q)$
  by(auto simp add: HInv4d-def)
with $disk'$
show $?thesis$
by(auto simp add: HInv4d-def)
qed

theorem HEndPhase2-HInv4d:
  $[\ HEndPhase2 \ s \ s' \ p; \ HInv4d \ s \ q] \Rightarrow H_{inv4d} \ s' \ q$
by(blast dest: HEndPhase2-HInv4d-p HEndPhase2-HInv4d-q)

lemma HFail-HInv4d-p:
  assumes act: $H_{fail} \ s \ s' \ p$
and inv: $H_{inv4d} \ s \ p$
shows $H_{inv4d} \ s' \ p$
proof(clarsimp simp add: HInv4d-def)
  fix $bk$
  assume $bk: bk \in blocksOf \ s' \ p$
  from act
  have $disk': disk \ s' = disk \ s$
    by(auto simp add: Fail-def)
  from subsetD[OF HFail-blocksOf[OF act] $bk$]
  show $\exists D \in MajoritySet. \ \forall d \in D. \ bal \ bk \leq mbal(disk \ s' \ d \ p)$
proof
  assume $bk: bk \in \{dblock \ s' \ p\}$
  with $inv \ disk'$
  show $?thesis$
    by(auto simp add: HInv4d-def)
next
  assume $bk: bk \in \{dblock \ s' \ p\}$
  with act
  have $bal \ bk = 0$
    by(auto simp add: Fail-def InitDB-def)
  with Disk-isMajority
  show $?thesis$
    by(auto simp add: MajoritySet-def)

86
lemma HFail-HInv4d-q:
assumes act: HFail s s' p
and inv: HInv4d s q
and pnq: p ≠ q
shows HInv4d s' q
proof –
  from act pnq
  have disk': disk s' = disk s
  and dblock': dblock s' q = dblock s q
    by (auto simp add: Fail-def)
  from act pnq
  have blocksRead': ∀ q. allRdBlks s' q ⊆ allRdBlks s q
    by (auto simp add: Fail-def InitializePhase-def allRdBlks-def)
  with disk' dblock'
  have blocksOf s' q ⊆ blocksOf s q
    by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  from subsetD[OF this] inv
  have ∀ bk ∈ blocksOf s' q.
    ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s d q)
    by (auto simp add: HInv4d-def)
  with disk'
  show ?thesis
    by (auto simp add: HInv4d-def)
qed

theorem HFail-HInv4d:
[ HFail s s' p; HInv4d s q ] ⇒ HInv4d s' q
by (blast dest: HFail-HInv4d-p HFail-HInv4d-q)

lemma HPhase0Read-HInv4d-p:
assumes act: HPhase0Read s s' p d
and inv: HInv4d s p
shows HInv4d s' p
proof(clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk ∈ blocksOf s' p
  from act
  have bal': bal (dblock s' p) = bal (dblock s p)
    by (auto simp add: Phase0Read-def)
  from subsetD[OF HPhase0Read-blocksOf[OF act] bk] inv
  have ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s d p)
    by (auto simp add: HInv4d-def)
  with act
  show ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s' d p)
    by (auto simp add: Phase0Read-def)
qed
lemma HPhase0Read-HInv4d-q:
assumes act: HPhase0Read s s' p d
and inv: HInv4d s q
and pnq: p≠q
shows HInv4d s' q
proof
  from act pnq
  have disk': disk s' = disk s
    by (auto simp add: Phase0Read-def)
  from act pnq
  have blocksOf s' q ⊆ blocksOf s q
    by (auto simp add: Phase0Read-def allRdBlks-def blocksOf-def rdBy-def)
  from subsetD[OF this] inv
  have ∀ bk∈blocksOf s' q.
    ∃ D∈MajoritySet. ∀ d∈D. bal bk ≤ mbal(disk s d q)
    by (auto simp add: HInv4d-def)
  thus ?thesis
  by (auto simp add: HInv4d-def)
qed

theorem HPhase0Read-HInv4d:
[ HPhase0Read s s' p d; HInv4d s q ] ⇒ HInv4d s' q
by (blast dest: HPhase0Read-HInv4d-p HPhase0Read-HInv4d-q)

lemma HEndPhase0-blocksOf2:
assumes act: HEndPhase0 s s' p
and inv2c: Inv2c-inner s p
shows allBlocksRead s p ⊆ blocksOf s p
proof
  from act inv2c
  have ∀ d.∀ br ∈ blocksRead s p d. proc br =p ∧ block br = disk s d p
    by (auto simp add: EndPhase0-def Inv2c-inner-def)
  thus ?thesis
  by (auto simp add: allBlocksRead-def allRdBlks-def blocksOf-def)
qed

lemma HEndPhase0-HInv4d-p:
assumes act: HEndPhase0 s s' p
and inv: HInv4d s p
and inv2c: Inv2c s
and inv1: Inv1 s
shows HInv4d s' p
proof(clarsimp simp add: HInv4-d-def)
  fix bk
assume \( bk : bk \in \text{blocksOf} \ s' \ p \)
from \( \text{subsetD}[\text{OF HEndPhase0-blocksOf}[\text{OF act}] \ bk] \)
have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal}(\text{disk} \ s \ d \ p) \)

proof
assume \( bk : bk \in \text{blocksOf} \ s \ p \)
with \( \text{inv} \)
show \( \text{?thesis} \)
  by(\( \text{auto simp add: HInv4d-def} \))
next
assume \( bk : bk \in \{ \text{dblock} \ s' \ p \} \)
from \( \text{inv2c} \)
have \( \text{inv2c-inner: Inv2c-inner} \ s \ p \)
  by(\( \text{auto simp add: Inv2c-def} \))
from \( \text{bk HEndPhase0-some}[\text{OF act inv1}] \)
  \( \text{HEndPhase0-blocksOf2}[\text{OF act inv2c-inner}] \ \text{act} \)
have \( \text{bal} \ bk \in \text{bal} ' (\text{blocksOf} \ s \ p) \)
  by(\( \text{auto simp add: EndPhase0-def} \))
with \( \text{inv} \)
show \( \text{?thesis} \)
  by(\( \text{auto simp add: HInv4d-def} \))
qed

with \( \text{act} \)
show \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal}(\text{disk} \ s' \ d \ p) \)
  by(\( \text{auto simp add: EndPhase0-def} \))
qed

lemma \( \text{HEndPhase0-HInv4d-q} : \)
assumes \( \text{act: HEndPhase0} \ s \ s' \ p \)
and \( \text{inv: HInv4d} \ s \ q \)
and \( \text{pnq: p \# q} \)
shows \( \text{HInv4d} \ s' \ q \)

proof
from \( \text{act pnq} \)
have \( \text{dblock} \ s' \ q = \text{dblock} \ s \ q \land \text{disk} \ s' = \text{disk} \ s \)
  by(\( \text{auto simp add: EndPhase0-def} \))
moreover
from \( \text{act pnq} \)
have \( \forall p \ d. \ \text{rdBy} \ s' \ q \ p \ d \subseteq \text{rdBy} \ s \ q \ p \ d \)
  by(\( \text{auto simp add: EndPhase0-def InitializePhase-def rdBy-def} \))

hence \( (\text{UN} p \ d. \ \text{rdBy} \ s' \ q \ p \ d) \subseteq (\text{UN} p \ d. \ \text{rdBy} \ s \ q \ p \ d) \)
  by(\( \text{auto, blast} \))
ultimately
have \( \text{blocksOf} \ s' \ q \subseteq \text{blocksOf} \ s \ q \)
  by(\( \text{auto simp add: blocksOf-def, blast} \))
from \( \text{subsetD}[\text{OF this}] \ \text{inv} \)
have \( \forall bk \in \text{blocksOf} \ s' \ q. \)
  \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal}(\text{disk} \ s \ d \ q) \)
  by(\( \text{auto simp add: HInv4d-def} \))
with \( \text{act} \)

89
**Show** \( ? \text{thesis} \)

by (auto simp add: EndPhase0-def HInv4d-def)

**Qed**

**Theorem** \( \text{HEndPhase0-HInv4d} \):

\[
[ \text{HEndPhase0} \ s \ s' \ p; \ \text{HInv4d} \ s \ q; \\
\text{Inv2c} \ s; \ \text{Inv1} \ s' ] \implies \text{HInv4d} \ s' \ q
\]

by (blast dest: HEndPhase0-HInv4d-p HEndPhase0-HInv4d-q)

Since we have already proved \( \text{HInv2} \) is an invariant of \( \text{HNext} \), \( \text{HInv1} \land \text{HInv2} \land \text{HInv4} \) is also an invariant of \( \text{HNext} \).

**Lemma I2d:**

assumes \( \text{nxt} : \text{HNext} \ s \ s' \)
and \( \text{inv} : \text{HInv1} \ s \land \text{HInv2} \ s \land \text{HInv2} \ s' \land \text{HInv4} \ s \)
shows \( \text{HInv4} \ s' \)

**Proof**

(auto simp add: HInv4-def)

fix \( p \)

**Show** \( \text{HInv4a} \ s' \ p \) using \( \text{assms} \)

by (auto simp add: HInv4-def HNext-def Next-def)

(auto simp add: HInv2-def intro: HStartBallot-HInv4a,
auto intro: HPhase0Read-HInv4a,
auto intro: HPhase1or2Write-HInv4a,
auto simp add: Phase1or2Read-def

intro: HPhase1or2ReadThen-HInv4a
HPhase1or2ReadElse-HInv4a,
auto simp add: EndPhase1or2-def
intro: HEndPhase1-HInv4a
HEndPhase2-HInv4a,
auto intro: HFail-HInv4a,
auto intro: HEndPhase0-HInv4a simp add: HInv1-def)

**Show** \( \text{HInv4b} \ s' \ p \) using \( \text{assms} \)

by (auto simp add: HInv4-def HNext-def Next-def,
auto simp add: HInv2-def

intro: HStartBallot-HInv4b,
auto intro: HPhase0Read-HInv4b,
auto intro: HPhase1or2Write-HInv4b,
auto simp add: Phase1or2Read-def

intro: HPhase1or2ReadThen-HInv4b
HPhase1or2ReadElse-HInv4b,
auto simp add: EndPhase1or2-def
intro: HEndPhase1-HInv4b
HEndPhase2-HInv4b,
auto intro: HFail-HInv4b,
auto intro: HEndPhase0-HInv4b simp add: HInv1-def Inv2c-def)

**Show** \( \text{HInv4c} \ s' \ p \) using \( \text{assms} \)

by (auto simp add: HInv4-def HNext-def Next-def,
auto simp add: HInv2-def

intro: HStartBallot-HInv4c,
auto intro: HPhase0Read-HInv4c,
auto intro: HPhase1or2Write-HInv4c, auto simp add: Phase1or2Read-def intro: HPhase1or2ReadThen-HInv4c HPhase1or2ReadElse-HInv4c, auto simp add: EndPhase1or2-def intro: HEndPhase1-HInv4c HEndPhase2-HInv4c, auto intro: HFail-HInv4c, auto intro: HEndPhase0-HInv4c simp add: HInv1-def)

show HInv4d s' p using assms by (auto simp add: HInv4-def HNext-def Next-def, auto simp add: HInv2-def intro: HStartBallot-HInv4d, auto intro: HPhase0Read-HInv4d, auto simp add: Phase1or2Read-def intro: HPhase1or2ReadThen-HInv4d HPhase1or2ReadElse-HInv4d, auto simp add: EndPhase1or2-def intro: HEndPhase1-HInv4d HEndPhase2-HInv4d, auto intro: HFail-HInv4d, auto intro: HEndPhase0-HInv4d simp add: HInv1-def)

qed
end

theory DiskPaxos-Inv5 imports DiskPaxos-Inv3 DiskPaxos-Inv4 begin

C.5 Invariant 5

This invariant asserts that, if a processor p is in phase 2, then either its bal and inp values satisfy maxBalInp, or else p must eventually abort its current ballot. Processor p will eventually abort its ballot if there is some processor q and majority set D such that p has not read q's block on any disk D, and all of those blocks have mbal values greater than bal(dblocksp).

definition maxBalInp :: state ⇒ nat ⇒ InputsOrNi ⇒ bool where maxBalInp s b v = (∀ bk∈allBlocks s. b ≤ bal bk → inp bk = v)

definition Hinv5-inner-R :: state ⇒ Proc ⇒ bool where Hinv5-inner-R s p = (maxBalInp s (bal(dblock s p)) (inp(dblock s p))

∨ (∃ D∈MajoritySet. ∃ q. (∀ d∈D. bal(dblock s p) < mbal(disk s d q) ∧ ¬hasRead s p d q)))

definition Hinv5-inner :: state ⇒ Proc ⇒ bool

91
where \( HInv5\text{-inner} \, s \, p = (phase \, s \, p = 2 \implies HInv5\text{-inner-R} \, s \, p) \)

definition \( HInv5 :: \text{state} \Rightarrow \text{bool} \)
\[
where \quad HInv5 \, s = (\forall \, p. \, HInv5\text{-inner} \, s \, p)
\]

C.5.1 Proof of Invariant 5

The initial state implies Invariant 5.

theorem \( HInit\text{-HInv5} : HInit \, s \implies HInv5 \, s \)
\[
\text{using} \quad \text{Disk-isMajority}
\]
\[
\text{by(auto simp add:HInit-def Init-def HInv5-def HInv5-inner-def)}
\]

We will use the notation used in the proofs of invariant 4, and prove the lemma \( \text{action-HInv5-p} \) and \( \text{action-HInv5-q} \) for each action, for the cases \( p = q \) and \( p \neq q \) respectively.

Also, for each action we will define an \( \text{action-allBlocks} \) lemma in the same way that we defined \( \text{-blocksOf} \) lemmas in the proofs of \( HInv2 \). Now we prove that for each action the new \( allBlocks \) are included in the old \( allBlocks \) or, in some cases, included in the old \( allBlocks \) union the new \( dblock \).

lemma \( HStartBallot\text{-HInv5-p} \):
\[
\text{assumes act: HStartBallot \, s \, s' \, p}
\]
\[
\text{and inv: HInv5\text{-inner} \, s \, p}
\]
\[
\text{shows HInv5\text{-inner} \, s' \, p using assms}
\]
\[
\text{by(auto simp add: StartBallot-def HInv5-inner-def)}
\]

lemma \( HStartBallot\text{-blocksOf-q} \):
\[
\text{assumes act: HStartBallot \, s \, s' \, p}
\]
\[
\text{and pq: p \neq q}
\]
\[
\text{shows blocksOf \, s' \, q \subseteq blocksOf \, s \, q using assms}
\]
\[
\text{by(auto simp add: StartBallot-def InitializePhase-def blocksOf-def rdBy-def)}
\]

lemma \( HStartBallot\text{-allBlocks} \):
\[
\text{assumes act: HStartBallot \, s \, s' \, p}
\]
\[
\text{shows allBlocks \, s' \subseteq allBlocks \, s \cup \{dblock \, s' \, p\}}
\]

proof
\[
\text{(auto simp del: HStartBallot-def simp add: allBlocks-def}
\]
\[
\text{dest: HStartBallot\text{-blocksOf-q[OF act]})}
\]

fix \( x \, pa \)
assume \( x\text{-pa}: x \in \text{blocksOf} \, s' \, pa \) and
\( x\text{-nblks}: \forall \, xa. \, x \notin \text{blocksOf} \, s \, xa \)
show \( x\text{-dblock} \, s' \, p \)

proof\( (\text{cases p=pa}) \)
\[
\text{case True}
\]
\[
\text{from x-nblks}
\]
\[
\text{have x \notin \text{blocksOf} \, s \, p}
\]
\[
\text{by auto}
\]
\[
\text{with True subsetD[OF HStartBallot\text{-blocksOf-q[OF act]} x-pa]}
\]

92
show ?thesis
by auto
next
case False
  from x-nblks subsetD[OF HStartBallot-blocksOf-q[OF act False] x-pa]
  show ?thesis
  by auto
qed
qed

lemma HStartBallot-HInv5-q1:
assumes act: HStartBallot s s’ p
and pnq: p ≠ q
and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q))
shows maxBalInp s’ (bal(dblock s’ q)) (inp(dblock s’ q))
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s’
  and bal: bal (dblock s’ q) ≤ bal bk
  from act pnq
  have dblock’: dblock s’ q = dblock s q
      by(auto simp add: StartBallot-def)
  from subsetD[OF HStartBallot-allBlocks[OF act] bk]
  show inp bk = inp (dblock s’ q)
proof
  assume bk: bk ∈ allBlocks s
  with inv5-1 dblock’ bal
  show ?thesis
      by(auto simp add: maxBalInp-def)
next
  assume bk: bk ∈ {dblock s’ p}
  have dblock s p ∈ allBlocks s
      by(auto simp add: allBlocks-def blocksOf-def)
  with bal act bk dblock’ inv5-1
  show ?thesis
      by(auto simp add: maxBalInp-def StartBallot-def)
qed
qed

lemma HStartBallot-HInv5-q2:
assumes act: HStartBallot s s’ p
and pnq: p ≠ q
and inv5-2: ∃D∈MajoritySet. ∃qq. (∀d∈D. bal(dblock s q) < mbal(disk s d qq)
                                 ∧ ~hasRead s q d qq)
shows ∃D∈MajoritySet. ∃qq. (∀d∈D. bal(dblock s’ q) < mbal(disk s’ d qq)
                                 ∧ ~hasRead s’ q d qq)
proof
  from act pnq
  have disk: disk s’ = disk s

93
and \( \text{blocksRead} : \forall d. \text{blocksRead} \ s \ s' \ q \ d = \text{blocksRead} \ s \ q \ d \)
and \( \text{dblock} : \text{dblock} \ s' \ q = \text{dblock} \ s \ q \)
by (auto simp add: StartBallot-def InitializePhase-def)

with \( \text{inv5-2} \)
show \(?\text{thesis}\) by (auto simp add: hasRead-def)
qed

lemma \( \text{HStartBallot-HInv5-q} \):
assumes \( \text{act} : \text{HStartBallot} \ s \ s' \ p \)
and \( \text{inv} : \text{HInv5-inner} \ s \ q \)
and \( \text{pnq} : p \neq q \)
shows \( \text{HInv5-inner} \ s' \ q \)
using \( \text{assms and HStartBallot-HInv5-q1[OF act pnq]} \) \( \text{HStartBallot-HInv5-q2[OF act pnq]} \)
by (auto simp add: HInv5-inner-def HInv5-inner-R-def StartBallot-def)

theorem \( \text{HStartBallot-HInv5} \):
\[ \text{HStartBallot} \ s \ s' \ p \quad \text{HInv5-inner} \ s \ q \implies \text{HInv5-inner} \ s' \ q \]
by (blast dest: HStartBallot-HInv5-q HStartBallot-HInv5-p)

lemma \( \text{HPhase1or2Write-HInv5-1} \):
assumes \( \text{act} : \text{HPhase1or2Write} \ s \ s' \ p \ d \)
and \( \text{inv5-1} : \text{maxBalInp} \ s (\text{bal}(\text{dblock} \ s \ q)) (\text{inp}(\text{dblock} \ s \ q)) \)
sows \( \text{maxBalInp} \ s' (\text{bal}(\text{dblock} \ s' \ q)) (\text{inp}(\text{dblock} \ s' \ q)) \)
using \( \text{assms and HPhase1or2Write-blocksOf[OF act]} \)
by (auto simp add: Phase1or2Write-def maxBalInp-def allBlocks-def)

lemma \( \text{HPhase1or2Write-HInv5-p2} \):
assumes \( \text{act} : \text{HPhase1or2Write} \ s \ s' \ p \ d \)
and \( \text{inv4c} : \text{HInv4c} \ s \ p \)
and \( \text{phase} : \text{phase} \ s \ p = 2 \)
and \( \text{inv5-2} : \exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock} \ s \ p) < \text{mbal}(\text{disk} \ s \ d \ q)) \)
\( \land \neg \text{hasRead} \ s \ p \ d \ q \)
shows \( \exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock} \ s' \ p) < \text{mbal}(\text{disk} \ s' \ d \ q)) \)
\( \land \neg \text{hasRead} \ s' \ p \ d \ q \)

proof –
from \( \text{inv5-2} \)
obtain \( D \) \( q \)
where \( \text{i1} : \text{IsMajority} \ D \)
and \( \text{i2} : \forall d \in D. \text{bal}(\text{dblock} \ s \ p) < \text{mbal}(\text{disk} \ s \ d \ q) \)
and \( \text{i3} : \forall d \in D. \neg \text{hasRead} \ s \ p \ d \ q \)
by (auto simp add: MajoritySet-def)

have \( \text{pnq} : p \neq q \)
proof –
from \( \text{inv4c phase} \)
obtain \( D1 \) where \( \text{r1} : \text{IsMajority} \ D1 \land (\forall d \in D1. \text{mbal}(\text{disk} \ s \ d \ p) = \text{bal}(\text{dblock} \ s \ p)) \)

by(auto simp add: HInv4c-def MajoritySet-def)
with 11 majorities-intersect
have $D \cap D_1 \neq \emptyset$ by auto
then obtain $dd$ where $dd \in D \cap D_1$
  by auto
thus thesis by auto
qed

from act pnq — dblock and hasRead do not change
have dblock $s' = dblock s$
  and $\forall d. hasRead s' p' d q = hasRead s p d q$
— In all disks $q$ blocks don’t change
  and $\forall d. disk s' d q = disk s d q$
  by(auto simp add: Phase1or2Write-def hasRead-def)
with 2 2 3 majority-nonempty
have $\forall d \in D. bal (dblock s' p) < mbal (disk s d q)$
  by(auto simp add: Phase1or2Write-def)
  and $\neg hasRead s' p d q$
  by auto
with 1
show thesis
  by(auto simp add: MajoritySet-def)
qed

lemma HPhase1or2Write-HInv5-p:
assumes act: HPhase1or2Write $s s' p d$
and inv: HInv5-inner $s p$
and inv4: HInv4c $s p$
shows HInv5-inner $s' p$
proof(auto simp add: HInv5-inner-def HInv5-inner-R-def)
  assume phase': $phase s' p = 2$
  and $\forall D \in MajoritySet. \forall q. \exists d \in D. bal (dblock s' p) < mbal (disk s d q)$
  $\rightarrow hasRead s' p d q$
  with act have phase: $phase s p = 2$
    by(auto simp add: Phase1or2Write-def)
  show maxBalInp $s' (bal (dblock s' p)) (inp (dblock s' p))$
    proof(OF HPhase1or2Write-HInv5-1[OF act, of p])
      from HPhase1or2Write-HInv5-p[OF act inv4 phase] inv i2 phase
      show maxBalInp $s (bal (dblock s p)) (inp (dblock s p))$
        by(auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
    qed
  qed

qed

lemma HPhase1or2Write-allBlocks:
assumes act: HPhase1or2Write $s s' p d$
shows allBlocks $s' \subseteq allBlocks s$
using HPhase1or2Write-blocksOf[OF act]
by(auto simp add: allBlocks-def)
lemma HPhase1or2Write-HInv5-q2:
  assumes act: HPhase1or2Write s s' p d and pnq: p\neq q and inv4a: HInv4a s p and inv5-2: \exists D\in MajoritySet. \exists qq. (\forall d\in D. bal(dblock s q) < mbal(disk s d qq) 
  \wedge \neg hasRead s q d qq)
  shows \exists D\in MajoritySet. \exists qq. (\forall d\in D. bal(dblock s' q) < mbal(disk s' d qq) 
  \wedge \neg hasRead s' q d qq)

proof –
  from inv5-2
  obtain D qq
    where i1: IsMajority D
    and i2: \forall d\in D. bal(dblock s q) < mbal(disk s d qq)
    and i3: \forall d\in D. \neg hasRead s q d qq
    by(auto simp add: MajoritySet-def)
  from act pnq
     — dblock and hasRead do not change
  have dblock': dblock s' = dblock s
    and hasread: \forall d. hasRead s' q d qq = hasRead s q d qq
    by(auto simp add: Phase1or2Write-def hasRead-def)
  have \forall d\in D. bal(dblock s' q) < mbal(disk s' d qq) \wedge \neg hasRead s' q d qq
  proof(cases qq=p)
    case True
    have bal(dblock s q) < mbal(dblock s p)
    proof –
      from inv4a act i1
      have \exists d\in D. mbal(disk s d p) \leq mbal(dblock s p)
      by(auto simp add: MajoritySet-def HInv4a-def HInv4a2-def Phase1or2Write-def)
      with True i2
      show bal(dblock s q) < mbal(dblock s p)
      by auto
    qed
    with hasread dblock' True i1 i2 i3 act
    show ?thesis
    by(auto simp add: Phase1or2Write-def)
  next
    case False
    with act i2 i3
    show ?thesis
    by(auto simp add: Phase1or2Write-def hasRead-def)
  qed
  with i1
  show ?thesis
  by(auto simp add: MajoritySet-def)
qed
lemma HPhase1or2Write-HInv5-q:
  assumes act: HPhase1or2Write s s' p d
  and inv: HInv5-inner s q
  and inv4a: HInv4a s p
  and pnq: p ≠ q
  shows HInv5-inner s' q
proof(auto simp add: HInv5-inner-def HInv5-inner-R-def)
  assume phase': phase s' q = 2
  from phase' act have phase: phase s q = 2
    by(auto simp add: Phase1or2Write-def)
  show maxBalInp s (bal (dblock s q)) (inp (dblock s q))
    proof(rule HPhase1or2Write-HInv5-1[OF act, of q])
    from HPhase1or2Write-HInv5-q2[OF act pnq inv4a] inv i2 phase
    show maxBalInp s (bal (dblock s q)) (inp (dblock s q))
      by(auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
  qed
qed

theorem HPhase1or2Write-HInv5:
  [ HPhase1or2Write s s' p d; HInv5-inner s q; HInv4c s p; HInv4a s p ] ==> HInv5-inner s' q
by(auto simp add: Phase1or2Write-HInv5-1[OF act], blast)

lemma HPhase1or2ReadThen-HInv5-1:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv5-1: maxBalInp s (bal (dblock s q)) (inp (dblock s q))
  shows maxBalInp s' (bal (dblock s' q)) (inp (dblock s' q))
using assms and HPhase1or2ReadThen-blocksOf[OF act]
by(auto simp add: Phase1or2ReadThen-def maxBalInp-def allBlocks-def)

lemma HPhase1or2ReadThen-HInv5-p2:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv4c: HInv4c s p
  and inv2c: Inv2c-inner s p
  and phase: phase s p = 2
  and inv5-2: ∃ D ∈ MajoritySet. ∃ q. (∀ d ∈ D. bal (dblock s p) < mbal (disk s d q) ∧ ~hasRead s p d q)
  shows ∃ D ∈ MajoritySet. ∃ q. (∀ d ∈ D. bal (dblock s' p) < mbal (disk s' d q) ∧ ~hasRead s' p d q)
proof
  from inv5-2 obtain D q
    where i1: IsMajority D
      and i2: ∀ d ∈ D. bal (dblock s p) < mbal (disk s d q)
      and i3: ∀ d ∈ D. ~hasRead s p d q
    by(auto simp add: MajoritySet-def)
  from inv2c phase
have \( \text{bal}(\text{dblock } s \ p) = \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: Inv2c-inner-def)
moreover
from \( \text{act} \) have \( \text{mbal} \ (\text{disk } s \ d \ r) < \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: Phase1or2ReadThen-def)
moreover
from \( \tilde{i} \tilde{2} \) have \( d \in D \rightarrow \text{bal}(\text{dblock } s \ p) < \text{mbal}(\text{disk } s \ d \ q) \) by auto
ultimately have \( \text{pn} \text{r: } d \in D \rightarrow q \neq r \) by auto
have \( \text{pn} \text{q: } p \neq q \)
proof –
from \( \text{inv}4c \) phase
obtain \( D1 \) where \( r1: \text{IsMajority } D1 \land (\forall d \in D1. \text{mbal}(\text{disk } s \ d \ p) = \text{bal}(\text{dblock } s \ p)) \)
by (auto simp add: HInv4c-def MajoritySet-def)
with \( \tilde{i} \tilde{1} \) majorities-intersect
have \( D \cap D1 \neq \{\} \) by auto
then obtain \( dd \) where \( dd \in D \cap D1 \)
by auto
with \( \tilde{i} \tilde{1} \tilde{2} \) \( r1 \)
have \( \text{bal}(\text{dblock } s \ p) < \text{mbal}(\text{disk } s \ dd \ q) \land \text{mbal}(\text{disk } s \ dd \ p) = \text{bal}(\text{dblock } s \ p) \)
by auto
thus \( \text{?thesis} \) by auto
qed
from \( \text{pn} \text{r act} \)
have \( \text{hasRead’}: \forall d \in D. \text{hasRead } s \ p \ d \ q = \text{hasRead } s \ p \ d \ q \)
by (auto simp add: Phase1or2ReadThen-def hasRead-def)
from \( \text{act} \) \( \text{pn} \)
— \( \text{dblock and disk do not change} \)
have \( \text{dblock } s’ = \text{dblock } s \)
and \( \forall d. \text{disk } s’ = \text{disk } s \)
by (auto simp add: Phase1or2ReadThen-def)
with \( \tilde{i} \tilde{2} \) \( \text{hasRead’} \) \( \tilde{i} \tilde{3} \)
have \( \forall d \in D. \text{bal} (\text{dblock } s’ \ p) < \text{mbal}(\text{disk } s’ \ d \ q) \land \neg \text{hasRead } s’ \ p \ d \ q \)
by auto
with \( \tilde{i} \)
show \( \text{?thesis} \)
by (auto simp add: MajoritySet-def)
qed

lemma \( \text{HPhase1or2ReadThen-HInv5-p:} \)
assumes \( \text{act: } \text{HPhase1or2ReadThen } s \ s’ \ p \ d \ r \)
and \( \text{inv: } \text{HInv5-inner } s \ p \)
and \( \text{inv}4: \text{HInv4c } s \ p \)
and \( \text{inv}2c: \text{Inv2c } s \)
shows \( \text{HInv5-inner } s’ \ p \)
proof (auto simp add: HInv5-inner-def HInv5-inner-R-def)
assume \( \text{phase’}: \text{phase } s’ \ p = 2 \)
and \( \tilde{i}2: \forall D \in \text{MajoritySet}. \forall q. \exists d \in D. \text{bal} (\text{dblock } s’ \ p) < \text{mbal}(\text{disk } s’ \ d \ q) \rightarrow \text{hasRead } s’ \ p \ d \ q \)

98
with act have phase: phase s p = 2
  by (auto simp add: Phase1or2ReadThen-def)

show maxBalInp s' (bal (dblock s' p)) (inp (dblock s' p))
proof (rule HPhase1or2ReadThen-HInv5-1 [OF act, of p])
  from inv2c
  have Inv2c-inner s p by (auto simp add: Inv2c-def)
  from HPhase1or2ReadThen-HInv5-p2 [OF act inv4 this phase]
  have maxBalInp s (bal (dblock s p)) (inp (dblock s p))
    by (auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
  qed

lemma HPhase1or2ReadThen-allBlocks:
  assumes act: HPhase1or2ReadThen s s' p d r
  shows allBlocks s' ⊆ allBlocks s
using HPhase1or2ReadThen-blocksOf [OF act]
proof (auto simp add: allBlocks-def)

lemma HPhase1or2ReadThen-HInv5-q2:
  assumes act: HPhase1or2ReadThen s s' p d r
  and pnq: p ≠ q
  and inv4a: HInv4a s p
  and inv5-2: ∃ D ∈ MajoritySet. ∃ qq. (∀ d ∈ D. bal (dblock s q) < mbal (disk s d qq))
    ∧ ¬ hasRead s q d qq
  shows ∃ D ∈ MajoritySet. ∃ qq. (∀ d ∈ D. bal (dblock s' q) < mbal (disk s' d qq))
    ∧ ¬ hasRead s' q d qq
proof
  from inv5-2
  obtain D q q where i1: IsMajority D
    and i2: ∀ d ∈ D. bal (dblock s q) < mbal (disk s d qq)
    and i3: ∀ d ∈ D. ¬ hasRead s q d qq
    by (auto simp add: MajoritySet-def)
  from act pnq
    have dblock': dblock s' = dblock s
    and disk': disk s' = disk s
    and hasread: ∀ d. hasRead s' q d qq = hasRead s q d qq
      by (auto simp add: Phase1or2ReadThen-def hasRead-def)
    with i2 i3
    have ∀ d ∈ D. bal (dblock s' q) < mbal (disk s' d qq) ∧ ¬ hasRead s' q d qq
      by auto
  with i1
  show thesis
    by (auto simp add: MajoritySet-def)
  qed

lemma HPhase1or2ReadThen-HInv5-q:
assumes act: HPhase1or2ReadThen s s' p d r
and inv: Hinv5-inner s q
and inv4a: Hinv4a s p
and pnq: p ≠ q
shows Hinv5-inner s' q
proof (auto simp add: Hinv5-inner-def Hinv5-inner-R-def)
  assume phase': phase s' q = 2
  and i2: \( \forall D \in \text{MajoritySet}. \forall qa. \exists d \in D. \text{bal} (\text{dblock} s' q) < \text{mbal} (\text{disk} s' d qa) \)
  \( \rightarrow \text{hasRead} s' q d qa \)
  from phase' act have phase: phase s q = 2
  by (auto simp add: Phase1or2ReadThen-def)
show maxBalInp s' (bal (dblock s' q)) (inp (dblock s' q))
proof (rule HPhase1or2ReadThen-HInv5-1 [OF act, of q])
  from HPhase1or2ReadThen-HInv5-q2 [OF act pnq inv4a] inv i2 phase
  show maxBalInp s (bal (dblock s q)) (inp (dblock s q))
  by (auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
qed

theorem HPhase1or2ReadThen-HInv5:
[ HPhase1or2ReadThen s s' p d r; Hinv5-inner s q; Inv2c s; Hinv4c s p; Hinv4a s p ] \( \Rightarrow \) Hinv5-inner s' q
by (blast dest: HPhase1or2ReadThen-HInv5-q HPhase1or2ReadThen-HInv5-p)

theorem HPhase1or2ReadElse-HInv5:
[ HPhase1or2ReadElse s s' p d r; Hinv5-inner s q ] \( \Rightarrow \) Hinv5-inner s' q
using HStartBallot-HInv5
by (auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase2-Hinv5-p:
HEndPhase2 s s' p \( \Rightarrow \) Hinv5-inner s' p
by (auto simp add: EndPhase2-def Hinv5-inner-def)

lemma HEndPhase2-allBlocks:
  assumes act: HEndPhase2 s s' p
  shows allBlocks s' \( \subseteq \) allBlocks s
  using HEndPhase2-blocksOf [OF act]
  by (auto simp add: allBlocks-def)

lemma HEndPhase2-Hinv5-q1:
  assumes act: HEndPhase2 s s' p
  and pnq: p ≠ q
  and inv5-1: maxBalInp s (bal (dblock s q)) (inp (dblock s q))
  shows maxBalInp s' (bal (dblock s' q)) (inp (dblock s' q))
proof (auto simp add: maxBalInp-def)
fix bk
assume bk: bk \( \in \) allBlocks s'
  and bal: bal (dblock s' q) ≤ bal bk

100
from act pnq
have dblock': dblock s' q = dblock s q by(auto simp add: EndPhase2-def)
from subsetD[OF HEndPhase2-allBlocks[OF act] bk] inv5-1 dblock' bal
show inp bk = inp (dblock s' q)
  by(auto simp add: maxBalInp-def)
qed

lemma HEndPhase2-HInv5-q2:
  assumes act: HEndPhase2 s s' p
  and pnq: p≠q
  and inv5-2: ∃ D∈MajoritySet. ∃ qq. (∀ d∈D. bal(dblock s q) < mbal(disk s d qq)
       ∧ ¬hasRead s q d qq)
  shows ∃ D∈MajoritySet. ∃ qq. (∀ d∈D. bal(dblock s' q) < mbal(disk s' d qq)
       ∧ ¬hasRead s' q d qq)
proof –
  from act pnq
  have disk: disk s' = disk s
  and blocksRead: ∀ d. blocksRead s' q d = blocksRead s q d
  and dblock: dblock s' q = dblock s q
  by(auto simp add: EndPhase2-def InitializePhase-def)
  with inv5-2
  show ?thesis
    by(auto simp add: hasRead-def)
qed

lemma HEndPhase2-HInv5-q:
  assumes act: HEndPhase2 s s' p
  and inv: HInv5-inner s q
  and pnq: p≠q
  shows HInv5-inner s' q
using assms and HEndPhase2-HInv5-q1[OF act pnq] HEndPhase2-HInv5-q2[OF act pnq]
by(auto simp add: HInv5-inner-def HInv5-inner-R-def EndPhase2-def)

theorem HEndPhase2-HInv5:
[ HEndPhase2 s s' p; HInv5-inner s q ] ⇒ HInv5-inner s' q
by(blast dest: HEndPhase2-HInv5-q HEndPhase2-HInv5-p)

lemma HEndPhase1-HInv5-p:
  assumes act: HEndPhase1 s s' p
  and inv4: HInv4 s
  and inv2a: Inv2a s
  and inv2b: Inv2a s'
  and inv2c: Inv2c s
  and asm4: ¬maxBalInp s' (bal(dblock s' p)) (inp(dblock s' p))
  shows (∃ D∈MajoritySet. ∃ q. (∀ d∈D. bal(dblock s' p) < mbal(disk s' d q)
        ∧ ¬hasRead s' p d q))
proof –

101
have \( \exists bk \in \text{allBlocks } s. \, \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p \)

proof -
from asm4
obtain bk
  where p31: \( bk \in \text{allBlocks } s' \land \, \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p \)
  by(auto simp add: maxBalImp-def)
then obtain q where p32: \( bk \in \text{blocksOf } s' q \)
  by(auto simp add: allBlocks-def)
from act
have dblock: \( p \neq q \implies \text{dblock } s' q = \text{dblock } s q \)
  by(auto simp add: EndPhase1-def)
have bk \( p \in \text{blocksOf } s q \)
proof(cases \( p = q \))
  case True
  with p32 p31 HEndPhase1-blocksOf[OF act]
  show ?thesis
    by auto
next
  case False
  from dblock[OF False] subsetD[OF HEndPhase1-blocksOf[OF act, of q] p32]
  show ?thesis
    by(auto simp add: blocksOf-def)
qed
with p31
show ?thesis
  by(auto simp add: allBlocks-def)
qed
then obtain bk where p22: \( bk \in \text{allBlocks } s \land \, \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p \)
  by(auto)
have \( \exists q \in \text{UNIV} - \{ p \} \cdot \, bk \in \text{blocksOf } s q \)
proof -
  from p22
  obtain q where bk: \( bk \in \text{blocksOf } s q \)
    by(auto simp add: allBlocks-def)
  from act p22
  have mbal(\text{dblock } s p) \leq \text{bal } bk
    by(auto simp add: EndPhase1-def)
moreover
from act
have phase s p = 1
  by(auto simp add: EndPhase1-def)
moreover
from inv4
have Hinv4b s p by(auto simp add: Hinv4-def)
ultimately
have p \( q \)
  using bk
  by(auto simp add: Hinv4-def Hinv4b-def)
with bk

102
show \(\forall\)thesis
by auto
qed
then obtain \(q\) where \(p23: q \in \text{UNIV} - \{p\} \land bk \in \text{blocksOf} s q\)
by auto
have \(\exists D \in \text{MajoritySet}. \forall d \in D. \: \text{bal}(dblock s' p) \leq \text{mbal}(disk s d q)\)
proof
from \(p23\) inv4
have \(i4d: \exists D \in \text{MajoritySet}. \forall d \in D. \: \text{bal} bk \leq \text{mbal}(disk s d q)\)
by (auto simp add: HInv4-def HInv4d-def)
from \(i4d\) p22
show \(\forall\)thesis
by force
qed
then obtain \(D\) where \(Dmaj: D \in \text{MajoritySet} \land p24: (\forall d \in D. \: \text{bal}(dblock s' p) \leq \text{mbal}(disk s d q))\)
by auto
have \(p25: (\forall d \in D. \: \text{bal}(dblock s' p) < \text{mbal}(disk s d q))\)
proof
from inv2c
have Inv2c-inner s p
by (auto simp add: Inv2c-def)
with act
have bal-pos: \(0 < \text{bal}(dblock s' p)\)
by (auto simp add: Inv2c-inner-def EndPhase1-def)
with inv2a
have bal(dblock s' p) \in \text{Ballot} p \cup \{0\}
by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
with bal-pos have bal-in-p: \(\text{bal}(dblock s' p) \in \text{Ballot} p\)
by auto
from inv2a have Inv2a-inner s q
by (auto simp add: Inv2a-def)
next have \(\forall d \in D. \: \text{mbal}(disk s d q) \in \text{Ballot} q \cup \{0\}\)
by (auto simp add: Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
with p24 bal-pos
have \(\forall d \in D. \: \text{mbal}(disk s d q) \in \text{Ballot} q\)
by force
with Ballot-disj p23 bal-in-p
have \(\forall d \in D. \: \text{mbal}(disk s d q) \neq \text{bal}(dblock s' p)\)
by force
with p23 p24
show \(\forall\)thesis
by force
qed
with \(p23\) act
have \(\forall d \in D. \: \text{bal}(dblock s' p) < \text{mbal}(disk s' d q) \land \neg \text{hasRead} s' p d q\)
by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def)
with \(Dmaj\)
show ?thesis
  by blast
qed

lemma union-inclusion:
  \[ A \subseteq A'; B \subseteq B' \] \implies A \cup B \subseteq A' \cup B'
by blast

lemma HEndPhase1-blocksOf-q:
  assumes act: HEndPhase1 s s' p
and pnq: p \neq q
shows blocksOf s' q \subseteq blocksOf s q
proof −
  from act pnq
  have dblock: \{dblock s' q\} \subseteq \{dblock s q\}
  and disk: disk s' = disk s
  and blks: blocksRead s' q = blocksRead s q
  by(auto simp add: HEndPhase1-def InitializePhase-def)
  from disk
  have disk': \{disk s' d q | d \in UNIV\} \subseteq \{disk s d q | d \in UNIV\} (is ?D)
  by auto
  from pnq act
  have (\UN qq d. rdBy s' q qq d) \subseteq (\UN qq d. rdBy s q qq d)
  by(auto simp add: HEndPhase1-def InitializePhase-def rdBy-def split: split-if-asm, blast)
  hence \{block br | br. br \in (\UN qq d. rdBy s' q qq d)\} \subseteq \{block br | br. br \in (\UN qq d. rdBy s q qq d)\} (is ?R)
  by blast
  from union-inclusion[OF dblock union-inclusion[OF disk' this]]
  show ?thesis
  by(auto simp add: blocksOf-def)
qed

lemma HEndPhase1-allBlocks:
  assumes act: HEndPhase1 s s' p
shows allBlocks s' \subseteq allBlocks s \cup \{dblock s' p\}
proof(auto simp del: HEndPhase1-def simp add: allBlocks-def
  dest: HEndPhase1-blocksOf-q[OF act])
  fix x pa
  assume x-pa: x \in blocksOf s' pa and
  x-nblks: \forall xa. x \notin blocksOf s xa
  show x = dblock s' p
proof(cases p = pa)
  case True
  from x-nblks
  have x \notin blocksOf s p
  by auto
  with True subsetD[OF HEndPhase1-blocksOf[OF act] x-pa]
show \(\exists\)thesis
by auto
next
case False
from x-nblks subsetD[OF HEndPhase1-blocksOf-q[OF act False] x-pa]
show \(\exists\)thesis
by auto
qed
qed

lemma HEndPhase1-HInv5-q:
assumes act: HEndPhase1 s s' p
and inv: HInv5 s
and inv1: Inv1 s
and inv2a: Inv2a s'
and inv2a-q: Inv2a s
and inv2b: Inv2b s
and inv2c: Inv2c s
and inv3: HInv3 s
and phase': phase s' q = 2
and pnq: p\#q
and asm4': \neg maxBalInp s' (bal(dblock s' q)) (inp(dblock s' q))
shows (\exists D\in MajoritySet. \exists q q. (\forall d\in D. bal(dblock s' q) < mbal(disk s d qq)
\land \neg hasRead s' q d qq))

proof –
from act pnq
have phase s' q = phase s q
and phase-p: phase s p = 1
and disk: disk s' = disk s
and dblock: dblock s' q = dblock s q
and bal: bal(dblock s' p) = mbal(dblock s p)
by(auto simp add: EndPhase1-def InitializePhase-def)
with phase'
have phase: phase s q = 2 by auto
from phase inv2c
have bal-dblk-q: bal(dblock s q) \in Ballot q
by(auto simp add: Inv2c-def Inv2c-inner-def)
have \exists D\in MajoritySet. \exists q q. (\forall d\in D. bal(dblock s q) < mbal(disk s d qq)
\land \neg hasRead s q d qq)
proof(cases maxBalInp s (bal(dblock s q)) (inp(dblock s q)))
case True
have p21: bal(dblock s q) < bal(dblock s' p) \land inp(dblock s q) \# inp(dblock s' p)
proof –
from True asm4 dblock HEndPhase1-allBlocks[OF act]
have p32: bal(dblock s q) \leq bal(dblock s' p)
\land inp(dblock s q) \# inp(dblock s' p)
by(auto simp add: maxBalInp-def)
from inv2a
105
have \( \text{bal}(\text{dblock } s' \ p) \in \text{Ballot } p \cup \{ \theta \} \) 
\hspace{1cm} \text{by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)}

moreover 
from \( \text{Ballot-disj Ballot-nzero pnq} \)
have \( \text{Ballot } q \cap (\text{Ballot } p \cup \{ \theta \}) = \{ \} \) 
\hspace{1cm} \text{by auto}

ultimately 
have \( \text{bal}(\text{dblock } s' \ p) \neq \text{bal}(\text{dblock } s \ q) \) 
\hspace{1cm} \text{using bal-dblk-q} 
\hspace{1cm} \text{by auto}

ultimately 
have \( \exists \, D \in \text{MajoritySet} \forall \, d \in D. \text{bal}(\text{dblock } s \ q) < \text{mbal}(\text{disk } s \ d \ p) \land \text{hasRead } s \ p \ d \ q \) 
\hspace{1cm} \text{proof} 
\hspace{1cm} \text{from act} 
\hspace{1cm} have \( \exists \, D \in \text{MajoritySet} : \forall \, d \in D. \, \text{d in disksWritten } s \ p \land (\forall \, q \in \text{UNIV} - \{ p \}, \text{hasRead } s \ p \ d \ q) \) 
\hspace{1cm} \hspace{1cm} \text{by (auto simp add: EndPhase1-def MajoritySet-def)}

then obtain \( D \)
\hspace{1cm} \text{where} \( \text{act1: } \forall \, d \in D. \, \text{d in disksWritten } s \ p \land (\forall \, q \in \text{UNIV} - \{ p \}, \text{hasRead } s \ p \ d \ q) \)
\hspace{1cm} \hspace{1cm} \text{and} \( \text{Dmaj: } D \in \text{MajoritySet} \)
\hspace{1cm} \hspace{1cm} \hspace{1cm} \text{by auto}

from inv2b 
have \( \forall \, d. \, \text{Inv2b-inner } s \ p \ d \ q \) \hspace{1cm} \text{by (auto simp add: Inv2b-def)}
\hspace{1cm} \text{with act1 pnq phase-p bal}
\hspace{1cm} have \( \forall \, d \in D. \, \text{bal}(\text{dblock } s' \ p) = \text{mbal}(\text{disk } s \ d \ p) \land \text{hasRead } s \ p \ d \ q \)
\hspace{1cm} \hspace{1cm} \text{by (auto simp add: Inv2b-def Inv2b-inner-def)}

with p21 Dmaj 
have \( \forall \, d \in D. \, \text{bal}(\text{dblock } s \ q) < \text{mbal}(\text{disk } s \ d \ p) \land \text{hasRead } s \ p \ d \ q \)
\hspace{1cm} \hspace{1cm} \text{by auto}
\hspace{1cm} \hspace{1cm} \text{with Dmaj}
\hspace{1cm} \hspace{1cm} \hspace{1cm} \text{show } \text{?thesis} 
\hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \text{by auto}

\hspace{1cm} \hspace{1cm} \text{qed}

then obtain \( D \)
\hspace{1cm} \text{where} \( p22: \, D \in \text{MajoritySet} \land (\forall \, d \in D. \, \text{bal}(\text{dblock } s \ q) < \text{mbal}(\text{disk } s \ d \ p) \land \text{hasRead } s \ p \ d \ q) \)
\hspace{1cm} \hspace{1cm} \text{by auto}

have \( p23: \, \forall \, d \in D. \, \text{dblock-\{d in allBlocksRead } s \ p \land (\forall \, q \in \text{blocksRead } s \ p \ d \ q) \)
\hspace{1cm} \hspace{1cm} \text{proof} 
\hspace{1cm} \hspace{1cm} \text{have } \text{dblock } q \in \text{allBlocksRead } s \ p \land (\forall \, q \in \text{blocksRead } s \ p \ d \ q) \text{proof} 
\hspace{1cm} \hspace{1cm} \hspace{1cm} \text{assume } \text{dblock-q: } \text{dblock } s \ q \in \text{allBlocksRead } s \ p \)
\hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \text{from inv2a-q}
have \((\text{bal}(\text{dblock} \ s \ q) = 0) = (\text{inp}(\text{dblock} \ s \ q) = \text{NotAnInput})\) by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def Inv2a-innermost-def)

with \(\text{bal-dblk-q} \ \text{Ballot-nzero} \ \text{dblock-q} \ \text{InputsOrNi}\)

have \(\text{dblock-q-nib}: \text{dblock} \ s \ q \in \text{nonInitBlks} \ s \ p\) by (auto simp add: nonInitBlks-def blocksSeen-def)

with \(\text{act}\)

have \(\text{dblock-max}: \text{inp}(\text{dblock} \ s' \ p) = \text{inp}(\text{maxBlk} \ s \ p)\) by (auto simp add: EndPhase1-def)

from \(\text{maxBlk-in-nonInitBlks[OF dblock-q-nib inv1]}\)

have \(\text{max-in-nib}: \text{maxBlk} \ s \ p \in \text{nonInitBlks} \ s \ p\)

hence \(\text{nonInitBlks} \ s \ p \subseteq \text{allBlocks} \ s\)

by (auto simp add: allBlocks-def nonInitBlks-def blocksSeen-def blocksOf-def rdBy-def)

with True subsetD[OF this max-in-nib]

have \(\text{bal} \ (\text{dblock} \ s \ q) \leq \text{bal} \ (\text{maxBlk} \ s \ p) \rightarrow \text{inp}(\text{maxBlk} \ s \ p) = \text{inp}(\text{dblock} \ s \ q)\) by (auto simp add: maxBalInp-def)

with maxBlk-in-nonInitBlks[OF dblock-q-nib inv1]

show \(\text{inp}(\text{dblock} \ s' \ p) = \text{inp}(\text{dblock} \ s \ q)\)

by auto

qed

with \(p21\)

have \(\forall d. \forall \text{br} \in \text{blocksRead} \ s \ q \ d. \text{bal}(\text{dblock} \ s \ q) \leq \text{mbal}(\text{block} \ \text{br})\)

proof

from inv2c phase

have \(\forall d. \forall \text{br} \in \text{blocksRead} \ s \ q \ d. \text{mbal}(\text{block} \ \text{br}) < \text{mbal}(\text{dblock} \ s \ q)\)

and \(\text{bal}(\text{dblock} \ s \ q) = \text{mbal}(\text{dblock} \ s \ q)\)

by (auto simp add: Inv2c-def Inv2c-inner-def)

thus \(?\text{thesis}\)

by force

qed

have \(p24: \forall d \in D. \neg(\exists \text{br} \in \text{blocksRead} \ s \ q \ d. \text{bal}(\text{dblock} \ s \ q) \leq \text{mbal}(\text{block} \ \text{br}))\)

proof

from inv2c phase

have \(\forall d. \forall \text{br} \in \text{blocksRead} \ s \ q \ d. \text{mbal}(\text{block} \ \text{br}) < \text{mbal}(\text{dblock} \ s \ q)\)

and \(\text{bal}(\text{dblock} \ s \ q) = \text{mbal}(\text{dblock} \ s \ q)\)

by (auto simp add: Inv2c-def Inv2c-inner-def)

thus \(?\text{thesis}\)

by force

qed

have \(p25: \forall d \in D. \neg\text{hasRead} \ s \ q \ d \ p\)

proof auto

fix \(d\)

assume \(d-in-D: d \in D\)

and \(\text{hasRead-qdp}: \text{hasRead} \ s \ q \ d \ p\)

have \(p31: (\text{block}=\text{dblock} \ s \ p, \text{proc}=\text{p}) \in \text{blocksRead} \ s \ q \ d\)

proof

from \(d-in-D \ p22\)


have hasRead-pdq; hasRead s p d q by auto
with hasRead-qdp phase phase-p inv3
have HInv3-R s q p d
  by (auto simp add: HInv3-def HInv3-inner-def HInv3-L-def)
with p23 d-in-D
show ?thesis
  by (auto simp add: HInv3-R-def)
qed

from p21 act
have p32: \(\text{bal}(\text{dblock s q}) < \text{mbal}(\text{dblock s p})\)
  by (auto simp add: EndPhase1-def)
with p31 d-in-D hasRead-qdp p24
show False
  by (force)
qed

with p22
show ?thesis
  by auto
next
  case False
  with inv phase
  show ?thesis
    by (auto simp add: HInv5-def HInv5-inner-def HInv5-inner-R-def)
qed

then obtain D qq
  where D \(\in\) MajoritySet \(\land\) (\(\forall d \in D\). \(\text{bal}(\text{dblock s q}) < \text{mbal}(\text{disk s d qq})\) \(\land\) \(\neg\) hasRead s q d qq)
    by auto
moreover
  from act pnq
  have \(\forall d. \text{hasRead s'} q d qq = \text{hasRead s q d qq}\)
    by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def)
ultimately show ?thesis
  using disk dblock
  by auto
qed


lemma HFail-HInv5-p:
  HFail s s' p \implies HInv5-inner s' p
by (auto simp add: HInv5-def HInv5-inner-def HInv5-inner-R-def)

lemma HFail-blocksOf-q:
  assumes act: HFail s s' p
  and p\not=q
  shows blocksOf s' q \subseteq blocksOf s q
using assms
by (auto simp add: Fail-def InitializePhase-def blocksOf-def rdBy-def)

lemma HFail-allBlocks:
  assumes act: HFail s s' p
  shows allBlocks s' \subseteq allBlocks s \cup \{dblock s' p\}
proof (auto simp del: HFail-def simp add: allBlocks-def)
  fix x pa
  assume x-pa: x \in blocksOf s' pa and
  x-nblks: \forall xa. x \not\in blocksOf s xa
  show x=dblock s' p
proof (cases p=pa)
  case True
  from x-nblks
  have x \not\in blocksOf s p
  by auto
  with True subsetD[OF HFail-blocksOf-q[OF act] x-pa]
  show ?thesis
  by auto
  next
  case False
  from x-nblks subsetD[OF HFail-blocksOf-q[OF act False] x-pa]
  show ?thesis
  by auto
qed

lemma HFail-HInv5-q1:
  assumes act: HFail s s' p
  and p\not=q
  and inv2a: Inv2a-inner s' q
  and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q))
  shows maxBalInp s' (bal(dblock s' q)) (inp(dblock s' q))
proof (auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk \in allBlocks s'
  and bal: bal (dblock s' q) \leq bal bk

qed
from act pnq
have dblock': dblock s' q = dblock s q by (auto simp add: Fail-def)
from subsetD[OF HFail-allBlocks[OF act] bk]
show inp bk = inp (dblock s' q)
proof
  assume bk: bk ∈ allBlocks s
  with inv5-1 dblock': bal
  show ?thesis
    by (auto simp add: maxBalInp-def)
next
  assume bk: bk ∈ {dblock s' p}
  with act have bk-init: bk = InitDB
  with bal
  have bal (dblock s' q) = 0
    by (auto simp add: InitDB-def)
  with inv2a
  have inp (dblock s' q) = NotAnInput
    by (auto simp add: Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
  with bk-init
  show ?thesis
    by (auto simp add: InitDB-def)
qed

lemma HFail-HInv5-q2:
  assumes act: HFail s s' p
  and pnq: p ≠ q
  and inv5-2: \( \exists D \in \text{MajoritySet}. \exists \ q q. (\forall d \in D. \ bal (\text{dblock s q}) < m\text{bal (disk s d q q)}) \wedge \neg \text{hasRead s q d q q}) \)
  shows \( \exists D \in \text{MajoritySet}. \exists \ q q. (\forall d \in D. \ bal (\text{dblock s' q}) < m\text{bal (disk s' d q q)}) \wedge \neg \text{hasRead s' q d q q}) \)
proof
  from act pnq
  have disk: disk s' = disk s
    and blocksRead: \( \forall d. \ \text{blocksRead s' q d} = \text{blocksRead s q d} \)
    and dblock: dblock s' q = dblock s q
    by (auto simp add: Fail-def InitializePhase-def)
  with inv5-2
  show ?thesis
    by (auto simp add: hasRead-def)
qed

lemma HFail-HInv5-q:
  assumes act: HFail s s' p
  and inv: HInv5-inner s q
  and pnq: p ≠ q
  and inv2a: Inv2a-inner s'

shows \( \text{HInv5-inner s'} q \)

proof(auto simp add: HInv5-inner-def HInv5-inner-R-def)

assume phase': phase s' q = 2

and nR2: \( \forall D \in \text{MajoritySet}. \forall qa. \exists d \in D. \text{bal (dblock s'} q) < m\text{bal (disk s' d qa) } \rightarrow \text{hasRead s'} q d qa (\text{is } ?P s') \)

from HFail-HInv5-q2[OF act pnq]

have \( \neg (\neg P s) \Longrightarrow \neg (\neg P s') \)

by auto

with nR2

have P: \( ?P s \)

by blast

from inv2a

have inv2a': Inv2a-inner s' q by (auto simp add: Inv2a-def)

from act pnq phase'

have phase s q = 2

by(auto simp add: Fail-def split: split-if_asm)

with inv HFail-HInv5-q1[OF act pnq inv2a'] P

show maxBalInp s' (\text{bal (dblock s'} q)) (\text{inp (dblock s'} q))

by(auto simp add: HInv5-inner-def HInv5-inner-R-def)

qed

theorem HFail-HInv5:

\[
\begin{align*}
[ \text{HFail s s'} p ; \text{HInv5-inner s q} ; \text{Inv2a s'} ] & \Longrightarrow \text{HInv5-inner s'} q \\
\end{align*}
\]

by(blast dest: HFail-HInv5-q HFail-HInv5-p)

lemma HPhase0Read-HInv5-p:

HPhase0Read s s' p d \Longrightarrow \text{HInv5-inner s'} p

by(auto simp add: Phase0Read-def HInv5-inner-def)

lemma HPhase0Read-allBlocks:

assumes act: HPhase0Read s s' p d

shows allBlocks s' \( \subseteq \) allBlocks s

using HPhase0Read-blocksOf[OF act]

by(auto simp add: allBlocks-def)

lemma HPhase0Read-HInv5-1:

assumes act: HPhase0Read s s' p d

and inv5-1: maxBalInp s (\text{bal (dblock s q)}) (\text{inp (dblock s q)})

shows maxBalInp s' (\text{bal (dblock s'} q)) (\text{inp (dblock s'} q))

using assms and HPhase0Read-blocksOf[OF act]

by(auto simp add: Phase0Read-def maxBalInp-def allBlocks-def)

lemma HPhase0Read-HInv5-q2:

assumes act: HPhase0Read s s' p d

and pnq: p\( \neq q \)

and inv5-2: \( \exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \text{bal (dblock s q}) < m\text{bal (disk s d qq)} \)

\( \wedge \neg \text{hasRead s q d qq} \)
\[\exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \quad \text{bal}(\text{dblock} s' q) < \text{mbal}(\text{disk} s' d q) \land \neg \text{hasRead} s' q d q)\]

**proof** –
- **from** act pnq
- **have** disk s' = disk s
  - **and** blocksRead: \(\forall d. \text{blocksRead} s' q d = \text{blocksRead} s q d\)
  - **and** dblock: \(\text{dblock} s' q = \text{dblock} s q\)
- **by**(auto simp add: Phase0Read-def InitializePhase-def)
- **with** inv5-2
- **show** \(?\text{thesis}\)
  - **by**(auto simp add: hasRead-def)
- **qed**

**lemma** HPhase0Read-HInv5-q:
  **assumes** act: HPhase0Read s s' p d
  **and** inv: \(H\text{inv}_5\)-inner s q
  **and** pnq: p \(\neq\) q
  **shows** \(H\text{inv}_5\)-inner s' q

**proof**(auto simp add: \(H\text{inv}_5\)-inner-def \(H\text{inv}_5\)-inner-R-def)
- **assume** phase': phase s' q = 2
  - **and** i2: \(\forall D \in \text{MajoritySet}. \forall q a. \exists d \in D. \text{bal}(\text{dblock} s' q d q) < \text{mbal}(\text{disk} s' d q a)\)
  - \(\rightarrow\) hasRead s' q d qa
  - **from** phase' act **have** phase: phase s q = 2
    - **by**(auto simp add: Phase0Read-def)
  - **show** maxBalInp s' (\(\text{bal}(\text{dblock} s' q)\)) (\(\text{inp}(\text{dblock} s' q)\))
    - **proof**(rule HPhase0Read-HInv5-1[OF act, of q])
      - **from** HPhase0Read-HInv5-q2[OF act pnq] inv i2 phase
      - **show** maxBalInp s (\(\text{bal}(\text{dblock} s q)\)) (\(\text{inp}(\text{dblock} s q)\))
        - **by**(auto simp add: \(H\text{inv}_5\)-inner-def \(H\text{inv}_5\)-inner-R-def, blast)
  - **qed**
- **qed**

**theorem** HPhase0Read-HInv5:
  \[\left[ H\text{Phase0Read} s s' p d; \ \text{Hinv}_5\text{-inner} s q \right] \implies \text{Hinv}_5\text{-inner} s' q\]
**by**(blast dest: HPhase0Read-HInv5-q HPhase0Read-HInv5-p)

**lemma** HEndPhase0-HInv5-p:
  HEndPhase0 s s' p \(\implies\) \(H\text{inv}_5\)-inner s' p
**by**(auto simp add: EndPhase0-def \(H\text{inv}_5\)-inner-def)

**lemma** HEndPhase0-blocksOf-q:
  **assumes** act: HEndPhase0 s s' p
  **and** pnq: p \(\neq\) q
  **shows** blocksOf s' q \(\subseteq\) blocksOf s q

**proof** –
- **from** act pnq
  - **have** dblock: \(\{\text{dblock} s' q\} \subseteq \{\text{dblock} s q\}\)
  - **and** disk: \(\text{disk} s' = \text{disk} s\)

112
\[ \text{and blks: blocksRead s' q = blocksRead s q} \]

by (auto simp add: EndPhase0-def InitializePhase-def)

from disk

have disk': \{ disk s' d q | d \in UNIV \} \subseteq \{ disk s d q | d \in UNIV \} \ (\text{is } ?D' \subseteq ?D) 
  by auto

from pnq act

have \{ \text{UN qq d. rdBy s' q qq d} \} \subseteq \{ \text{UN qq d. rdBy s q qq d} \} 
  by (auto simp add: EndPhase0-def InitializePhase-def rdBy-def split: split_if_asm, blast)

hence \{ \text{block br | br. br \in (UN qq d. rdBy s' q qq d)} \} \subseteq \{ \text{block br | br. br \in (UN qq d. rdBy s q qq d)} \} 
  by blast

from union-inclusion[of dblock union-inclusion[of disk' this]]

show ?thesis 
  by (auto simp add: blocksOf-def)

qed

lemma HEndPhase0-allBlocks:
  assumes act: HEndPhase0 s s' p 
  shows allBlocks s' \subseteq allBlocks s \cup \{ dblock s' p \} 
proof (auto simp del: HEndPhase0-def simp add: allBlocks-def dest: HEndPhase0-blocksOf-q[of act])

fix x pa
assume x-pa: \( x \in \text{blocksOf s' pa} \) \ and
  x-nblks: \( \forall xa. x \notin \text{blocksOf s xa} \)

show x=dblock s' p 
proof (cases p=pa)

  case True
  from x-nblks

  have x \notin \text{blocksOf s p} 
    by auto

  with True subsetD[of HEndPhase0-blocksOf[of act] x-pa]

  show ?thesis 
    by auto

next

  case False
  from x-nblks subsetD[of HEndPhase0-blocksOf-q[of act False] x-pa]

  show ?thesis 
    by auto

qed

qed

lemma HEndPhase0-HInv5-q1:
  assumes act: HEndPhase0 s s' p 
  and pnq: p\#q 
  and inv1: Inv1 s 
  and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q)) 

113
shows \( \text{maxBalInp } s \) (bal(dblock s' q)) \((\text{inp(dblock s' q)})\)

**proof**

(auto simp add: maxBalInp-def)

fix \( bk \)

assume \( bk : bk \in \text{allBlocks } s' \)

and \( \text{bal : bal (dblock s' q) } \leq \text{bal } bk \)

from \( \text{act } pnq \)

have \( \text{dblock }' : \text{dblock s' q } = \text{dblock s q } \)

by(auto simp add: EndPhase0-def)

from \( \text{subsetD[OF HEndPhase0-allBlocks[OF act] bk] } \)

show \( \text{inp bk } = \text{inp (dblock s' q)} \)

**proof**

assume \( bk : bk \in \text{allBlocks } s \)

with \( \text{inv5-1 dblock }' \)

show \( ?\text{thesis} \)

by(auto simp add: maxBalInp-def)

next

assume \( bk : bk \in \{ \text{dblock s' p } \} \)

with \( \text{HEndPhase0-some[OF act inv1] act} \)

have \( \exists \text{ba }\in \text{allBlocksRead s } p. \text{bal } \text{ba } = \text{bal (dblock s' p) } \land \text{inp } \text{ba } = \text{inp (dblock s' p)} \)

by(auto simp add: EndPhase0-def)

then obtain \( \text{ba} \)

where \( \text{ba-blksread : ba}\in \text{allBlocksRead s } p \)

and \( \text{ba-balinp : bal } \text{ba } = \text{bal (dblock s' p) } \land \text{inp } \text{ba } = \text{inp (dblock s' p)} \)

by auto

have \( \text{allBlocksRead s } p \subseteq \text{allBlocks s} \)

by(auto simp add: allBlocksRead-def allRdBlks-def

allBlocks-def blocksOf-def rdBy-def)

from \( \text{subsetD[OF this ba-blksread] ba-balinp} \text{bal bk dblock }' \text{inv5-1 } \)

show \( ?\text{thesis} \)

by(auto simp add: maxBalInp-def)

qed

**lemma HEndPhase0-HInv5-q2:**

assumes \( \text{act : HEndPhase0 } s \) s' p

and \( \text{pnq : p} \neq q \)

and \( \text{inv5-2 : } \exists D \in \text{MajoritySet. } \exists \text{qq. } (\forall d \in D. \text{bal(dblock s q) } < \text{mbal(disk s d qq)}) \land \neg \text{hasRead s q d qq) } \)

shows \( \exists D \in \text{MajoritySet. } \exists \text{qq. } (\forall d \in D. \text{bal(dblock s' q) } < \text{mbal(disk s' d qq)}) \land \neg \text{hasRead s' q d qq) } \)

**proof**

from \( \text{act pnq} \)

have \( \text{disk: disk s' } = \text{disk } s \)

and \( \text{blocksRead: } \forall d. \text{blocksRead s' } q d = \text{blocksRead s } q d \)

and \( \text{dblock: dblock s' q } = \text{dblock s q} \)

by(auto simp add: EndPhase0-def InitializePhase-def)

with \( \text{inv5-2} \)

show \( ?\text{thesis} \)
by (auto simp add: hasRead-def)

qed

lemma HEndPhase0-HInv5-q:
  assumes act: HEndPhase0 s s' p
  and inv: HInv5-inner s q
  and inv1: Inv1 s
  and pnq: p ≠ q
  shows HInv5-inner s' q
  using assms and
    HEndPhase0-HInv5-q1 [OF act pnq inv1]
    HEndPhase0-HInv5-q2 [OF act pnq]
  by (auto simp add: HInv5-inner-def HInv5-inner-R-def EndPhase0-def)

theorem HEndPhase0-HInv5:
[ HEndPhase0 s s' p; HInv5-inner s q; Inv1 s ] ⇒ HInv5-inner s' q
by (blast dest: HEndPhase0-HInv5-q HEndPhase0-HInv5-p)

HInv1 ∧ HInv2 ∧ HInv3 ∧ HInv4 ∧ HInv5 is an invariant of HNext.

lemma I2c:
  assumes nxt: HNext s s'
  and inv: HInv1 s ∧ HInv2 s ∧ HInv2 s' ∧ HInv3 s ∧ HInv4 s ∧ HInv5 s
  shows HInv5 s'
  using assms
  by (auto simp add: HInv5-def HNext-def Next-def,
      auto simp add: HInv2-def intro: HStartBallot-HInv5,
      auto intro: HPhase0Read-HInv5,
      auto simp add: HInv4-def intro: HPhase1or2Write-HInv5,
      auto simp add: Phase1or2Read-def
        intro: HPhase1or2ReadThen-HInv5
        HPhase1or2ReadElse-HInv5,
      auto simp add: EndPhase1or2-def HInv1-def HInv4-def HInv5-def
        intro: HEndPhase1-HInv5
        HEndPhase2-HInv5,
      auto intro: HFail-HInv5,
      auto intro: HEndPhase0-HInv5 simp add: HInv1-def)

end

theory DiskPaxos-Chosen imports DiskPaxos-Inv5 begin

C.6 Lemma I2f

To prove the final conjunct we will use the predicate valueChosen(v). This predicate is true if v is the only possible value that can be chosen as output. It also asserts that, for every disk d in D, if q has already read disksdp, then it has read a block with bal field at least b.
definition valueChosen :: state ⇒ InputsOrNi ⇒ bool
where
valueChosen s v =
(∃ b ∈ (UN p. Ballot p).
maxBalInp s b v
∧ (∃ p. ∃ D ∈ MajoritySet. (∀ d ∈ D. b ≤ bal(disk s d p)
∧ (∀ q. (phase s q = 1
∧ b ≤ mbal(dblock s q)
∧ hasRead s q d p)
→ (∃ br ∈ blocksRead s q d. b ≤ bal(block br))))
))

lemma HEndPhase1-valueChosen-inp:
assumes act: HEndPhase1 s s' q
and inv2a: Inv2a s
and asm1: b ∈ (UN p. Ballot p)
and bk-blocksOf: bk ∈ blocksOf s r
and bk: bk ∈ blocksSeen s q
and b-bal: b ≤ bal bk
and asm3: maxBalInp s b v
and inv1: Inv1 s
shows inp(dblock s' q) = v

proof −
from bk-blocksOf inv2a
have inv2a-bk: Inv2a-innermost s r bk
  by(auto simp add: Inv2a-def Inv2a-inner-def)
from Ballot-nzero asm1
have 0 < b by auto
with b-bal
have 0 < bal bk by auto
with inv2a-bk
have inp bk ≠ NotAnInput
  by(auto simp add: Inv2a-innermost-def)
with bk InputsOrNi
have bk-noninit: bk ∈ nonInitBlks s q
  by(auto simp add: nonInitBlks-def blocksSeen-def
       allBlocksRead-def allRdBlks-def)
with maxBlk-in-nonInitBlks[OF this inv1] b-bal
have maxBlk-b: b ≤ bal(maxBlk s q)
  by auto
from maxBlk-in-nonInitBlks[OF bk-noninit inv1]
have ∃ p d. maxBlk s q ∈ blocksSeen s p
  by(auto simp add: nonInitBlks-def blocksSeen-def)
  hence ∃ p. maxBlk s q ∈ blocksOf s p
    by(auto simp add: blocksOf-def blocksSeen-def
         allBlocksRead-def allRdBlks-def rdBy-def, force)
with maxBlk-b asm3
have inp(maxBlk s q) = v
  by(auto simp add: maxBalInp-def allBlocks-def)
with bk-noninit act
show ?thesis
  by (auto simp add: EndPhase1-def)
qed

lemma HEndPhase1-maxBalInp:
  assumes act: HEndPhase1 s \ s’ q
  and asm1: b \in (UN p. Ballot p)
  and asm2: D\in MajoritySet
  and asm3: maxBalInp s b v
  and asm4: \forall d\in D. b \leq bal(disk s d p)
       \land (\forall q. (\phase s q = 1
       \land b \leq mbal(dblock s q)
       \land hasRead s q d p
       ) \longrightarrow (\exists br \in blocksRead s q d. b \leq bal(block br)))
  and inv1: Inv1 s
  and inv2a: Inv2a s
  and inv2b: Inv2b s
  shows maxBalInp s’ b v
proof (cases \exists b \leq mbal(dblock s q))
case True
show ?thesis
proof (cases \exists p \neq q)
  assume \exists p \neq q
  have \exists d\in D. hasRead s q d p
  proof
    from act
    have IsMajority({d. d \in disksWritten s q \land (\forall r \in \text{UNIV} - \{q\} . hasRead s q d r})) (is IsMajority(?M))
      by (auto simp add: EndPhase1-def)
    with majorities-intersect asm2
    have \exists \forall q\in \text{UNIV} - \{q\} . hasRead s q d r
      by (auto simp add: MajoritySet-def)
    hence \exists d\in D. \forall r\in \text{UNIV} - \{q\} . hasRead s q d r
      by auto
    with \exists p \neq q
    show ?thesis
      by auto
qed
then obtain d where p41: \exists d\in D . hasRead s q d p
  by auto
with asm4 asm3 act True
have p42: \exists br \in blocksRead s q d . b \leq bal(block br)
  by (auto simp add: EndPhase1-def)
from True act
have thesis-L: b \leq bal (dblock s’ q)
  by (auto simp add: EndPhase1-def)
from p42
have \exists v(bblock s' q) = v
proof auto

fix br
assume br: br ∈ blocksRead s q d
and b-bal: b ≤ bal (block br)
hence br-rdBy: br ∈ (UN q d. rdBy s (proc br) q d)
  by(auto simp add: rdBy-def)
hence br-blksof: block br ∈ blocksOf s (proc br)
  by(auto simp add: blocksOf-def)
from br have br-bseen: block br ∈ blocksSeen s q
  by(auto simp add: blocksSeen-def allBlocksRead-def allRdBlks-def)
from HEndPhase1-valueChosen-imp[OF act inv2a asm1 br-blksof br-bseen b-bal asm3 inv1]
  show ?thesis .
qed

next
case False
from asm4 have p41: ∀d∈D. b ≤ bal (disk s d p)
  by auto
have p42: ∃d∈D. disk s d p = dblock s p
proof 
  from act have IsMajority {d. d∈disksWritten s q ∧ (∀p∈UNIV − {q}. hasRead s q d p)} (is IsMajority ?S)
    by(auto simp add: EndPhase1-def)
  with majorities-intersect asm2 have D ∩ ?S ≠ {}
    by(auto simp add: MajoritySet-def)
  hence ∃d∈D. d∈disksWritten s q
    by auto
  with inv2b False show ?thesis
    by(auto simp add: Inv2b-def Inv2b-inner-def)
qed
have inp(dblock s' q) = v
proof 
  from p42 p41 False have b-bal: b ≤ bal (dblock s q) by auto
  have db-blksof: (dblock s q) ∈ blocksOf s q
    by(auto simp add: blocksOf-def)
  have db-bseen: (dblock s q) ∈ blocksSeen s q
    by(auto simp add: blocksSeen-def)
  from HEndPhase1-valueChosen-imp[OF act inv2a asm1 db-blksof db-bseen b-bal asm3 inv1]
    show ?thesis .
qed
with asm3 HEndPhase1-allBlocks[OF act]
show ?thesis
  by(auto simp add: maxBalInp-def)
qed
next
case False
have dblock s' q ∈ allBlocks s'
  by(auto simp add: allBlocks-def blocksOf-def)
show ?thesis
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s
  and b-bal: b ≤ bal bk
  from subsetD[OF HEndPhase1-allBlocks[OF act] bk]
  show inp bk = v
  proof
    assume bk: bk ∈ allBlocks s
    with asm3 b-bal
    show ?thesis
      by(auto simp add: maxBalInp-def)
  next
    assume bk: bk ∈ {dblock s' q}
    from act False
    have ¬ b ≤ bal (dblock s' q)
      by(auto simp add: EndPhase1-def)
    with bk b-bal
    show ?thesis
      by(auto)
  qed
  qed
qed

lemma HEndPhase1-valueChosen2:
assumes act: HEndPhase1 s s' q
and asm4: ∀ d∈D. b ≤ bal(disk s d p)
  ∧ (∀ q. (phase s q = 1
       ∧ b ≤ mbal(dblock s q)
       ∧ hasRead s q d p
             ) → (∃ br∈blocksRead s q d. b ≤ bal(block br))) (is ?P s)
shows ?P s'
proof(auto)
  fix d
  assume d: d∈D
  with act asm4
  show b ≤ bal (disk s' d p)
    by(auto simp add: EndPhase1-def)
  fix d q
  assume d: d∈D
  and phase': phase s' q = Suc 0

119
and dblk-mbal: \( b \leq \text{mbal}(\text{dblock } s' q) \)

with \( \text{act} \)

have \( p31: \text{phase } s q = 1 \)
  and \( p32: \text{dblock } s' q = \text{dblock } s q \)
  by (auto simp add: EndPhase1-def split: split-if-asm)

with \( \text{dblk-mbal} \)

have \( b \leq \text{mbal}(\text{dblock } s q) \) by auto

moreover

assume hasRead: hasRead \( s' q d p \)

with \( \text{act} \)

have hasRead \( s q d p \)
  by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def split: split-if-asm)

ultimately

have \( \exists br \in \text{blocksRead } s q d. b \leq \text{bal}(\text{block } br) \)
  using \( p31, \text{asm4 } d \)
  by blast

with \( \text{act} \) hasRead

show \( \exists br \in \text{blocksRead } s' q d. b \leq \text{bal}(\text{block } br) \)
  by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def)

qed

\begin{proof}

\begin{itemize}
  \item[1] \textbf{theorem HEndPhase1-valueChosen:}
  \begin{itemize}
    \item assumes \( \text{act}: \text{HEndPhase1 } s s' q \)
    \item and \( \text{vc}: \text{valueChosen } s v \)
    \item and \( \text{inv1}: \text{Inv1 } s \)
    \item and \( \text{inv2a}: \text{Inv2a } s \)
    \item and \( \text{inv2b}: \text{Inv2b } s \)
    \item and \( \text{v-input}: v \in \text{Inputs} \)
    \item shows \( \text{valueChosen } s' v \)
  \end{itemize}

\end{itemize}

\begin{itemize}
  \item[2] \textbf{proof –}
  \begin{itemize}
    \item from \( \text{vc} \)
    \begin{itemize}
      \item obtain \( b, p, D \) where
        \begin{itemize}
          \item \( \text{asm1}: b \in (\text{UN } p, \text{Ballot } p) \)
          \item \( \text{asm2}: D \in \text{MajoritySet} \)
          \item \( \text{asm3}: \text{maxBalInp } s b v \)
          \item \( \text{asm4}: \forall d \in D. b \leq \text{bal}(\text{disk } s d p) \)
        \end{itemize}
        \begin{itemize}
          \item \( \land (\forall q. \text{phase } s q = 1 \land b \leq \text{mbal}(\text{dblock } s q) \land \text{hasRead } s q d p) \)
          \item \( \rightarrow (\exists br \in \text{blocksRead } s q d. b \leq \text{bal}(\text{block } br)) \)
        \end{itemize}
        by (auto simp add: valueChosen-def)
    \end{itemize}

\end{itemize}

\begin{itemize}
  \item[3] \textbf{from HEndPhase1-maxBalInp[OF \( \text{act } \) \( \text{asm1 } \) \( \text{asm2 } \) \( \text{asm3 } \) \( \text{asm4 } \) \( \text{inv1 } \) \( \text{inv2a } \) \( \text{inv2b } \) ]}
  \begin{itemize}
    \item have \( \text{maxBalInp } s' b v \).
  \end{itemize}

\end{itemize}

\begin{itemize}
  \item[4] \textbf{with HEndPhase1-valueChosen2[OF \( \text{act } \) \( \text{asm4 } \) ] \( \text{asm1 } \) \( \text{asm2 } \)}
  \begin{itemize}
    \item show \( \text{thesis} \)
    \begin{itemize}
      \item by (auto simp add: valueChosen-def)
    \end{itemize}
  \end{itemize}

\end{itemize}

\end{itemize}

\end{proof}

\begin{flushright}
\textbf{qed}
\end{flushright}
lemma HStartBallot-maxBalInp:
assumes act: HStartBallot s s' q
  and asm3: maxBalInp s b v
shows maxBalInp s' b v
proof (auto simp add: maxBalInp-def)
fix bk
assume bk: bk ∈ allBlocks s'
  and b-bal: b ≤ bal bk
from subsetD[OF HStartBallot-allBlocks[OF act] bk]
show inp bk = v
proof
  assume bk: bk ∈ allBlocks s'
  with asm3 b-bal
  show ?thesis
    by (auto simp add: maxBalInp-def)
next
assume bk: bk ∈ {dblock s' q}
from asm3
have b ≤ bal (dblock s q) ⇒ inp (dblock s q) = v
  by (auto simp add: maxBalInp-def allBlocks-def blocksOf-def)
with act bk b-bal
show ?thesis
  by (auto simp add: StartBallot-def)
qed

lemma HStartBallot-valueChosen2:
assumes act: HStartBallot s s' q
  and asm4: ∀d∈D. b ≤ bal (disk s d p)
  ∧ (∀q.( phase s q = 1
      ∧ b ≤ mbal (dblock s q)
      ∧ hasRead s q d p
      ) → (∃br∈blocksRead s q d. b ≤ bal (block br))) (is ?P s)
shows ?P s'
proof (auto)
fix d
assume d: d ∈ D
with act asm4
show b ≤ bal (disk s' d p)
  by (auto simp add: StartBallot-def)
fix d q
assume d: d ∈ D
  and phase': phase s' q = Suc 0
  and dblk-mbal: b ≤ mbal (dblock s' q)
  and hasRead: hasRead s' q d p
from phase' act hasRead
have p31: phase s q = 1
  and p32: dblock s' q = dblock s q
  by (auto simp add: StartBallot-def InitializePhase-def)
```isar
  \text{hasRead-def} \text{ split : split-if-asm)

  with \text{dblk-mbal}
  \text{have} \ b \leq \text{mbal(dblock} \ s \ q) \text{ by auto
  moreover
  from \text{act hasRead}
  \text{have} \ \text{hasRead} \ s \ q \ d \ p
    \text{ by(} \text{auto simp add: StartBallot-def InitializePhase-def}
    \text{ hasRead-def split: split-if-asm))

  \text{ultimately}
  \text{have} \ \exists \ br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal(block} \ br)
    \text{ using } p\text{?I asm4 d}
    \text{ by blast
  with \text{act hasRead}
  \text{show} \ \exists \ br \in \text{blocksRead} \ s' \ q \ d. \ b \leq \text{bal(block} \ br)
    \text{ by(} \text{auto simp add: StartBallot-def InitializePhase-def}
    \text{ hasRead-def))

  \text{qed}

  \text{theorem HStartBallot-valueChosen:}
  \text{assumes act: HStartBallot} \ s \ s' \ q
  \text{ and \ vc: valueChosen} \ s \ v
  \text{ and \ v-input: } v \in \text{Inputs}
  \text{shows valueChosen} \ s' \ v
  \text{proof --}
    \text{from \text{vc}
      \text{obtain} \ b\ p\ D \text{ where}
        \text{asm1: } b \in (\text{UN p. Ballot} \ p)
        \text{ and \ asm2: } D\in\text{MajoritySet}
        \text{ and \ asm3: } \text{maxBalInp} \ s \ b \ v
        \text{ and \ asm4: } \forall \ d \in D. \ b \leq \text{bal(disk} \ s \ d \ p)
          \wedge (\forall q. (\text{phase} \ s \ q = 1
          \wedge b \leq \text{mbal(dblock} \ s \ q)
          \wedge \text{hasRead} \ s \ q \ d \ p)
            \rightarrow (\exists br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal(block} \ br)))
            \text{ by(} \text{auto simp add: valueChosen-def})
      \text{from HStartBallot-maxBalInp[OF \text{act asm3]}
      \text{have maxBalInp} \ s' \ b \ v
        \text{. with HStartBallot-valueChosen2[OF \text{act asm4]} \text{asm1 asm2}
        \text{show \ \emph{thesis}
          \text{ by(} \text{auto simp add: valueChosen-def})

  \text{qed}

  \text{lemma HPhase1or2Write-maxBalInp:}
  \text{assumes act: HPhase1or2Write} \ s \ s' \ q \ d
    \text{ and \ asm3: } \text{maxBalInp} \ s \ b \ v
  \text{shows maxBalInp} \ s' \ b \ v
  \text{proof(} \text{auto simp add: maxBalInp-def})
    \text{fix } bk
    \text{assume } bk: \ bk \in \text{allBlocks} \ s'
```

and \( b \)-bal: \( b \leq \text{bal} \) \( bk \)
from \( \text{subsetD}[(\text{OF} \ H\text{Phase1or2Write-allBlocks}[\text{OF} \ \text{act}] \ bk)] \) \( \text{asm3} \) \( b \)-bal
show \( \text{inp} \ bk = v \)
by \((\text{auto simp add: maxBalInp-def})\)
qed

lemma \( H\text{Phase1or2Write-valueChosen2} \):
assumes \( \text{act}: H\text{Phase1or2Write} \ s \ s' \ pp \ d \)
and \( \text{asm2}: D \in \text{MajoritySet} \)
and \( \text{asm4}: \forall \ d \in D. \ b \leq \text{bal} \ (\text{disk} \ s \ d \ p) \)
\( \wedge (\forall \ q. (\text{phase} \ s \ q = 1 \wedge b \leq \text{mbal} \ (\text{dblock} \ s \ q)) \wedge \text{hasRead} \ s \ q \ d \ p)) \rightarrow (\exists \ br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal} (\text{block} \ br)) \) (is \( \text{?P} \ s\))
and \( \text{inv4}: H\text{inv4a} \ s \ pp \)
shows \( \text{?P} \ s' \)
proof \((\text{auto})\)
fix \( d1 \)
assume \( d: d1 \in D \)
show \( b \leq \text{bal} \ (\text{disk} \ s' \ d1 \ p) \)
proof \((\text{cases} \ d1 = d \wedge pp = p)\)
 case True
with \( \text{inv4} \) \( \text{act} \)
have \( H\text{inv4a2} \ s \ p \)
by \((\text{auto simp add: Phase1or2Write-def Hinv4a-def})\)
with \( \text{asm2} \) \( \text{majorities-intersect} \)
have \( \exists \ dd \in D. \ \text{bal} (\text{disk} \ s \ dd \ p) \leq \text{bal} (\text{dblock} \ s \ p) \)
by \((\text{auto simp add: Hinv4a2-def MajoritySet-def})\)
then obtain \( \text{dd} \) where \( \text{p41}: \ dd \in D \wedge \text{bal} (\text{disk} \ s \ dd \ p) \leq \text{bal} (\text{dblock} \ s \ p) \)
by \( \text{auto} \)
from \( \text{asm4} \) \( \text{p41} \)
have \( b \leq \text{bal} (\text{disk} \ s \ dd \ p) \)
by \( \text{auto} \)
with \( \text{p41} \)
have \( \text{p42}: b \leq \text{bal} (\text{dblock} \ s \ p) \)
by \( \text{auto} \)
from \( \text{act True} \)
have \( \text{dblock} \ s \ p = \text{disk} \ s' \ d \ p \)
by \((\text{auto simp add: Phase1or2Write-def})\)
with \( \text{p42} \) \( \text{True} \)
show \( ?\text{thesis} \)
by \( \text{auto} \)
next
case False
with \( \text{act} \) \( \text{asm4} \) \( d \)
show \( ?\text{thesis} \)
by \((\text{auto simp add: Phase1or2Write-def})\)
qed
next

123
fix $d \ q$

assume $d \in D$

and phase': phase $s' q = \text{Suc 0}$
and dblk-mbal: $b \leq \text{mbal} (\text{dblock}s'q)$
and hasRead: hasRead $s'qdp$

from phase' act hasRead

have p31: phase $s q = 1$
and p32: dblock $s' q = \text{dblock}s q$

by (auto simp add: Phase1or2Write-def InitializePhase-def
hasRead-def split : split-if-asm)

with dblk-mbal
have $b \leq \text{mbal}(\text{dblock}sq)$ by auto

moreover
from act hasRead
have hasRead $s q dp$

by (auto simp add: Phase1or2Write-def InitializePhase-def
hasRead-def split : split-if-asm)

ultimately
have $\exists br \in \text{blocksRead}s q dp. b \leq \text{bal}(\text{block} br)$

using p31 asm4 $d$

by blast

with act hasRead

show $\exists br \in \text{blocksRead}s q dp. b \leq \text{bal}(\text{block} br)$

by (auto simp add: Phase1or2Write-def InitializePhase-def
hasRead-def)

qed

theorem HPhase1or2Write-valueChosen:

assumes act: HPhase1or2Write $s s' q d$
and vc: valueChosen $s v$
and v-input: $v \in \text{Inputs}$
and inv4: HInv4a $s q$

shows valueChosen $s v$

proof --

from vc

obtain $b p D$ where

asm1: $b \in (\text{UN} p. \text{Ballot} p)$
and asm2: $D \in \text{MajoritySet}$
and asm3: $\text{maxBalInp}s b v$
and asm4: $\forall d \in D. \ b \leq \text{bal}(\text{disk}s dp)$

($\forall q. (\text{phase}s q = 1$

$\land b \leq \text{mbal}(\text{dblock}s q)$

$\land \text{hasRead}s q dp)$

$\rightarrow (\exists br \in \text{blocksRead}s q dp. b \leq \text{bal}(\text{block} br)))$

by (auto simp add: valueChosen-def)

from HPhase1or2Write-maxBalInp[OF act asm3]

have maxBalInp $s' b v$.

with HPhase1or2Write-valueChosen2[OF act asm2 asm4 inv4] asm1 asm2

show ?thesis
lemma \( HPhase1or2ReadThen\text{-}maxBalInp \):
assumes \( \text{act}: HPhase1or2ReadThen \ s \ s' q \ d \ p \)
and \( \text{asm3}: \text{maxBalInp} \ s \ b \ v \)
shows \( \text{maxBalInp} \ s' \ b \ v \)
proof (auto simp add: maxBalInp-def)

fix \( bk \)
assume \( bk: bk \in \text{allBlocks} \ s' \)
and \( b\text{-bal}: b \leq \text{bal} \ bk \)
from \( \text{subsetD}\[\text{OF} HPhase1or2ReadThen\text{-}allBlocks[\text{OF} \ \text{act}] \ \text{bk}] \ \text{asm3} b\text{-bal} \)
show \( \text{inp} \ bk = v \)
by (auto simp add: maxBalInp-def)

qed

lemma \( HPhase1or2ReadThen\text{-}valueChosen2 \):
assumes \( \text{act}: HPhase1or2ReadThen \ s \ s' q \ d \ pp \)
and \( \text{asm4}: \forall \ d \in D. \ b \leq \text{bal}(\text{disk} \ s \ d \ p) \)
and \( \forall \ q. (\text{phase} \ s \ q = 1 \land b \leq \text{mbal} (\text{dblock} \ s \ q) \land \text{hasRead} \ s \ q \ d \ p) \rightarrow (\exists br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal}(\text{block} \ br)) \) (is \( ?P \ s \))
shows \( ?P \ s' \)
proof (auto)

fix \( dd \)
assume \( d: dd \in D \)
with \( \text{act} \ \text{asm4} \)
show \( b \leq \text{bal} (\text{disk} \ s' \ dd \ p) \)
by (auto simp add: Phase1or2ReadThen-def)

fix \( dq \)
assume \( d: dd \in D \)
and \( \text{phase}'s: \text{phase} \ s' \ qq = \text{Suc} \ 0 \)
and \( \text{dblkm-bal}: b \leq \text{mbal} (\text{dblock} \ s' \ qq) \)
and \( \text{hasRead}: \text{hasRead} \ s' \ qq \ dd \ p \)
show \( \exists b r \in \text{blocksRead} \ s' \ qq \ dd. \ b \leq \text{bal} (\text{block} \ br) \)
proof (cases \( d=dd \land qq=q \land pp=p \))
case True
from \( d \ \text{asm4} \)
have \( b \leq \text{bal}(\text{disk} \ s \ dd \ p) \)
by auto
with \( \text{act} \ True \)
show \( ?thesis \)
by (auto simp add: Phase1or2ReadThen-def)
next
case False
with \( \text{phase}' \ \text{act} \)
have \( p31: \text{phase} \ s \ qq = 1 \)
and p32: dblock s' qq = dblock s qq
by(auto simp add: Phase1or2ReadThen-def)
with dblk-mbal
have b≤mbal(dblock s qq) by auto
moreover
from act hasRead False
have hasRead s qq dd p
by(auto simp add: Phase1or2ReadThen-def
hasRead-def split: split-if-asm)
ultimately
have ∃br∈blocksRead s qq dd. b≤bal(block br)
using p31 asm4 d
by blast
with act hasRead
show ∃br∈blocksRead s' qq dd. b≤bal(block br)
by(auto simp add: Phase1or2ReadThen-def hasRead-def)
qed

theorem HPhase1or2ReadThen-valueChosen:
assumes act: HPhase1or2ReadThen s s' q d p
and vc: valueChosen s v
and v-input: v ∈ Inputs
shows valueChosen s' v

proof –
from vc
obtain b p D where
  asm1: b ∈ (UN p. Ballot p)
  and asm2: D∈MajoritySet
  and asm3: maxBalInp s b v
  and asm4: ∀ d ∈ D. b ≤ bal(disk s d p)
    ∧ (∀ q. (phase s q = 1
      ∧ b ≤ mbal(dblock s q)
      ∧ hasRead s q d p)
        → (∃br∈blocksRead s q d. b ≤ bal(block br))))
  by(auto simp add: valueChosen-def)
from HPhase1or2ReadThen-maxBalInp[OF act asm3]
have maxBalInp s' b v .
with HPhase1or2ReadThen-valueChosen2[OF act asm4] asm1 asm2
show ?thesis
  by(auto simp add: valueChosen-def)
qed

theorem HPhase1or2ReadElse-valueChosen:
[ HPhase1or2ReadElse s s' p d r; valueChosen s v; v∈ Inputs ]
⇒ valueChosen s' v
using HStartBallot-valueChosen
by(auto simp add: Phase1or2ReadElse-def)
lemma HEndPhase2-maxBalInp:
  assumes act: HEndPhase2 s s' q
  and asm3: maxBalInp s b v
  shows maxBalInp s' b v
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s'
  and b-bal: b ≤ bal bk
  from subsetD[OF HEndPhase2-allBlocks[OF act] bk] asm3 b-bal
  show inp bk = v
    by(auto simp add: maxBalInp-def)
qed

lemma HEndPhase2-valueChosen2:
  assumes act: HEndPhase2 s s' q
  and asm4: ∀ d ∈ D. b ≤ bal(disk s d p)
    ∧ (∀ q.( phase s q = 1
    ∧ b ≤ mbal(dblock s q)
    ∧ hasRead s q d p
    ) → (∃ br ∈ blocksRead s q d. b ≤ bal(block br)))) (is ?P s)
  shows ?P s'
proof(auto)
  fix d
  assume d: d ∈ D
  with act asm4
  show b ≤ bal (disk s' d p)
    by(auto simp add: EndPhase2-def)
  fix d q
  assume d: d ∈ D
    and phase': phase s' q = Suc 0
    and dblk-mbal: b ≤ mbal (dblock s' q)
    and hasRead: hasRead s' q d p
  from phase' act hasRead
  have p31: phase s q = 1
    and p32: dblock s' q = dblock s q
    by(auto simp add: EndPhase2-def InitializePhase-def
    hasRead-def split : split-if-asm)
  with dblk-mbal
  have b ≤ mbal(dblock s q) by auto
  moreover
  from act hasRead
  have hasRead s q d p
    by(auto simp add: EndPhase2-def InitializePhase-def
    hasRead-def split : split-if-asm)
  ultimately
  have ∃ br ∈ blocksRead s q d. b ≤ bal(block br)
    using p31 asm4 d
    by blast
  with act hasRead
show \( \exists b \in \text{blocksRead} s' q d. \, b \leq \text{bal}(\text{block } b) \)

  by (auto simp add: EndPhase2-def InitializePhase-def hasRead-def)

qed

theorem HEndPhase2-valueChosen:
  assumes act: HEndPhase2 s s' q
  and vc: valueChosen s v
  and v-input: \( v \in \text{Inputs} \)
  shows valueChosen s' v
proof
  from vc
  obtain b p D where
    asm1: \( b \in (\bigcup p. \text{Ballot } p) \)
    and asm2: D \( \subseteq \text{MajoritySet} \)
    and asm3: maxBalInp s b v
    and asm4: \( \forall d \in D. \, b \leq \text{bal}(\text{disk } s d p) \)
    \( \land (\forall q. (\text{phase } s q = 1 \land b \leq \text{mbal}(\text{dblock } s q) \land \text{hasRead } s q d p) \rightarrow (\exists b' \in \text{blocksRead} s q d. \, b \leq \text{bal}(\text{block } b'))) \)
  by (auto simp add: valueChosen-def)
  from HEndPhase2-maxBalInp[OF act asm3]
  have maxBalInp s' b v.
  with HEndPhase2-valueChosen2[OF act asm4] asm1 asm2
  show ?thesis
  by (auto simp add: valueChosen-def)
qed

lemma HFail-maxBalInp:
  assumes act: HFail s s' q
  and asm1: \( b \in (\bigcup p. \text{Ballot } p) \)
  and asm3: maxBalInp s b v
  shows maxBalInp s' b v
proof (auto simp add: maxBalInp-def)
fix bk
assume bk: \( bk \in \text{allBlocks } s' \)
and b-bal: \( b \leq \text{bal } bk \)
from subsetD[OF HFail-allBlocks[OF act] bk]
show inp bk = v
proof
  assume bk: \( bk \in \text{allBlocks } s \)
  with asm3 b-bal
  show ?thesis
  by (auto simp add: maxBalInp-def)
next
assume bk: \( bk \in \{\text{dblock } s' q\} \)
with act
have bal bk = 0
by (auto simp add: Fail-def InitDB-def)
moreover
from Ballot-nzero asm1
have 0 < b
by auto
ultimately
show ?thesis
using b-bal
by auto
qed

lemma HFail-valueChosen2:
assumes act: HFail s s' q
and asm4: \( \forall d \in D. \ b \leq \text{bal}(\text{disk } s d p) \) \( \land (\forall q. (\text{phase } s q = 1 \land b \leq \text{mbal}(\text{dblock } s q) \land \text{hasRead } s q d p) \rightarrow (\exists br \in \text{blocksRead } s q d. b \leq \text{bal}(\text{block } br))) \) (is ?P s)
shows ?P s'
proof (auto)
fix d
assume d: d \in D
with act asm4
show b \leq \text{bal} (\text{disk } s' d p)
  by (auto simp add: Fail-def)
fix d q
assume d: d \in D
and phase': \text{phase } s' q = \text{Suc } 0
and dblk-mbal: b \leq \text{mbal} (\text{dblock } s' q)
and hasRead: \text{hasRead } s' q d p
from phase' act hasRead
have p31: phase s q = 1
and p32: \text{dblock } s' q = \text{dblock } s q
by (auto simp add: Fail-def InitializePhase-def hasRead-def split : split-if-asm)
with dblk-mbal
have b \leq \text{mbal} (\text{dblock } s q) by auto
moreover
from act hasRead
have hasRead s q d p
  by (auto simp add: Fail-def InitializePhase-def hasRead-def split : split-if-asm)
ultimately
have \( \exists br \in \text{blocksRead } s q d. b \leq \text{bal}(\text{block } br) \)
  using p31 asm4 d
  by blast
with act hasRead
show \( \exists br \in \text{blocksRead } s' q d. b \leq \text{bal}(\text{block } br) \)
by(auto simp add: Fail-def InitializePhase-def hasRead-def)

qed

definition HFail-valueChosen
assumes act: HFail s s' q
and vc: valueChosen s v
and v-input: v ∈ Inputs
shows valueChosen s' v

proof
from vc
obtain b p D where 
asm1: b ∈ (UN p. Ballot p)
and asm2: D ∈ MajoritySet
and asm3: maxBalInp s b v
and asm4: ∀ d ∈ D. b ≤ bal(disk s d p)
∧ (∀ q.( phase s q = 1 
∧ b ≤ mbal(dblock s q) 
∧ hasRead s d p ))
→ (∃ br ∈ blocksRead s a d. b ≤ bal(block br)))

by(auto simp add: valueChosen-def)
from HFail-maxBalInp[OF act asm1 asm3]
have maxBalInp s' b v.
with HFail-valueChosen2[OF act asm4] asm1 asm2
show ?thesis
by(auto simp add: valueChosen-def)

qed

definition HPhase0Read-maxBalInp:
assumes act: HPhase0Read s s' q d
and asm3: maxBalInp s b v
shows maxBalInp s' b v

proof(auto simp add: maxBalInp-def)
fix bk
assume bk: bk ∈ allBlocks s'
and b-bal: b ≤ bal bk
from subsetD[OF HPhase0Read-allBlocks[OF act] bk] asm3 b-bal
show inp bk = v
by(auto simp add: maxBalInp-def)

qed

definition HPhase0Read-valueChosen2:
assumes act: HPhase0Read s s' q q d d
and asm4: ∀ d ∈ D. b ≤ bal(disk s d p)
∧ (∀ q.( phase s q = 1 
∧ b ≤ mbal(dblock s q) 
∧ hasRead s d p ))
→ (∃ br ∈ blocksRead s a d. b ≤ bal(block br))) (is ?P s)

shows ?P s'

proof(auto)
fix $d$
assume $d: d \in D$
with act asm4
show $b \leq \text{bal}(\text{disk } s' d p)$
by(auto simp add: Phase0Read-def)

next
fix $d q$
assume $d: d \in D$
and phase': phase $s' q = \text{Suc 0}$
and dblk-mbal: $b \leq \text{mbal}(\text{dblock } s' q)$
and hasRead: hasRead $s' q d p$
from phase' act have qqnq: $qq \neq q$
by(auto simp add: Phase0Read-def)
show $\exists \text{br} \in \text{blocksRead } s' q d. b \leq \text{bal}(\text{block br})$
proof –
from phase' act hasRead have p31: phase $s q = 1$
and p32: dblock $s' q = \text{dblock } s q$
by(auto simp add: Phase0Read-def hasRead-def)
with dblk-mbal
have $b \leq \text{mbal}(\text{dblock } s q)$ by auto
moreover
from act hasRead qqnq have hasRead $s q d p$
by(auto simp add: Phase0Read-def hasRead-def
split: split-if-asm)
ultimately
have $\exists \text{br} \in \text{blocksRead } s q d. b \leq \text{bal}(\text{block br})$
using p31 asm4 d
by blast
with act hasRead
show $\exists \text{br} \in \text{blocksRead } s' q d. b \leq \text{bal}(\text{block br})$
by(auto simp add: Phase0Read-def InitializePhase-def
hasRead-def)
qed

theorem HPhase0Read-valueChosen:
assumes act: HPhase0Read $s s' q d$
and vc: valueChosen $s v$
and v-input: $v \in \text{Inputs}$
shows valueChosen $s' v$
proof –
from vc obtain $b p D$ where
asm1: $b \in (\text{UN } p. \text{Ballot } p)$
and asm2: $D \in \text{MajoritySet}$
and asm3: maxBalInp $s b v$

qed
and \(asm4\): \(\forall d \in D. \ b \leq bal(disk s d p)\)
\[\land \forall q. (\ phase s q = 1 \land b \leq mball(dblock s q) \land hasRead s q d p) \rightarrow (\exists br \in \text{blocksRead} s q d. b \leq bal(block br))\]

- by (auto simp add: valueChosen-def)
- from HPhase0Read-maxBalInp[of act asm3]
- have maxBalInp \(s' \ b \ v\).
- with HPhase0Read-valueChosen2[of act asm4] asm1 asm2
- show \(?thesis\)
  - by (auto simp add: valueChosen-def)
qed

lemma HEndPhase0-maxBalInp:
- assumes act: HEndPhase0 \(s s' q\)
- and asm3: maxBalInp \(s b v\)
- and inv1: Inv1 s
- shows maxBalInp \(s' b v\)
proof (auto simp add: maxBalInp-def)
  fix \(bk\)
  assume \(bk\): \(bk \in \text{allBlocks} s'\)
  - and \(b\)-bal: \(b \leq \text{bal} bk\)
  from subsetD[of HEndPhase0-allBlocks[of act] bk]
  show \(\text{inp} \ bk = v\)
  proof
    assume \(bk\): \(bk \in \text{allBlocks} s\)
    with asm3 \(b\)-bal
    show \(?thesis\)
      - by (auto simp add: maxBalInp-def)
  next
  assume \(bk\): \(bk \in \{dblock s' q\}\)
  with HEndPhase0-some[of act inv1] act
  have \(\exists ba \in \text{allBlocksRead} s q. \text{bal} ba = \text{bal} (dblock s' q) \land \text{inp} ba = \text{inp} (dblock s' q)\)
  - by (auto simp add: EndPhase0-def)
  then obtain \(ba\)
    where \(ba\)-blksread: \(ba \in \text{allBlocksRead} s q\)
    - and \(ba\)-balinp: \(bal ba = \text{bal} (dblock s' q) \land \text{inp} ba = \text{inp} (dblock s' q)\)
    by auto
  have allBlocksRead \(s q \subseteq \text{allBlocks} s\)
  - by (auto simp add: allBlocksRead-def allRdBlks-def
    allBlocks-def blocksOf-def rdBy-def)
  from subsetD[of this ba-blksread] ba-balinp \(bk \ b\)-bal asm3
  show \(?thesis\)
    - by (auto simp add: maxBalInp-def)
qed
lemma HEndPhase0-valueChosen2:
assumes act: HEndPhase0 s s' q
and asm4: \( \forall d \in D. \ b \leq \text{bal}(\text{disk} \ s \ d \ p) \)
\( \land (\forall q. (\ \text{phase} \ s \ q = 1 \)
\( \land b \leq \text{mbal}(\text{dblock} \ s \ q) \)
\( \land \text{hasRead} \ s \ q \ d \ p) \)
\( \rightarrow (\exists br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal}(\text{block} \ br)) \) (is \ ?P s)
shows ?P s'
proof (auto)
fix d
assume d: \( d \in D \)
with act asm4
show \( b \leq \text{bal}(\text{disk} \ s' \ d \ p) \)
by (auto simp add: EndPhase0-def)
fix d q
assume d: \( d \in D \)
and phase': phase s' q = Suc 0
and dblk-mbal: \( b \leq \text{mbal}(\text{dblock} \ s' \ q) \)
and hasRead: hasRead s' q d p
from phase' act hasRead
have p31: phase s q = 1
and p32: dblock s' q = dblock s q
by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def split : split-if-asm)
with dblk-mbal
have \( b \leq \text{mbal}(\text{dblock} \ s \ q) \) by auto
moreover
from act hasRead
have hasRead s q d p
by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def split: split-if-asm)
ultimately
have \( \exists br \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal}(\text{block} \ br) \)
using p31 asm4 d
by blast
with act hasRead
show \( \exists br \in \text{blocksRead} \ s' \ q \ d. \ b \leq \text{bal}(\text{block} \ br) \)
by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def)
qed

theorem HEndPhase0-valueChosen:
assumes act: HEndPhase0 s s' q
and vc: valueChosen s v
and v-input: \( v \in \text{Inputs} \)
and inv1: Inv1 s
shows valueChosen s' v
proof –
from vc
obtain $b \ p \ D$ where

asm1: $b \in (\text{UN } p \text{. Ballot } p)$

and asm2: $D \in \text{MajoritySet}$

and asm3: $\text{maxBalInp } s \ b \ v$

and asm4: $\forall d \in D. \ b \leq \text{bal}(\text{disk } s \ d \ p)$

$\land (\forall q. (\text{phase } s \ q = 1$

$\land b \leq \text{mbal}(\text{dblock } s \ q)$

$\land \text{hasRead } s \ q \ d \ p)$

$\rightarrow (\exists br \in \text{blocksRead } s \ q \ d. \ b \leq \text{bal}(\text{block } br)))$

by (auto simp add: valueChosen-def)

from $\text{HEndPhase0-maxBalInp}[\text{OF } act \ \text{asm3 } \text{inv1}]$

have $\text{maxBalInp } s' \ b \ v$

with $\text{HEndPhase0-valueChosen2}[\text{OF } act \ \text{asm4}]$ $\text{asm1 } \text{asm2}$

show $?\text{thesis}$

by (auto simp add: valueChosen-def)

qed


theory DiskPaxos-Inv6 imports DiskPaxos-Chosen begin

C.7 Invariant 6

The final conjunct of $HInv$ asserts that, once an output has been chosen, $\text{valueChosen}(\text{chosen})$ holds, and each processor’s output equals either chosen or NotAnInput.

definition $HInv6 :: \text{state } \Rightarrow \text{bool}$

where

$HInv6 \ s = ((\text{chosen } s \neq \text{NotAnInput} \rightarrow \text{valueChosen } s \ (\text{chosen } s))$

$\land (\forall p. \text{outpt } s \ p \in \{\text{chosen } s, \text{NotAnInput}\}))$

theorem $HInit-HInv6$: $HInit \ s \Rightarrow HInv6 \ s$

by (auto simp add: HInit-def Init-def InitDB-def HInv6-def)

lemma $\text{HEndPhase2-Inv6-1}$:

assumes act: $\text{HEndPhase2 } s \ s' \ p$

and inv: $HInv6 \ s$

and inv2b: $\text{Inv2b } s$

and inv2c: $\text{Inv2c } s$

and inv3: $HInv3 \ s$

and inv5: $HInv5-inner \ s \ p$

and chosen': $\text{chosen } s' \neq \text{NotAnInput}$

shows $\text{valueChosen } s' \ (\text{chosen } s')$

proof (cases $\text{chosen } s = \text{NotAnInput}$)

from inv5 act

have inv5R: $HInv5-inner-R \ s \ p$

and phase: $\text{phase } s \ p = 2$

and ep2-maj: $\text{IsMajority } \{d. \ d \in \text{disksWritten } s \ p$
\( (\forall q \in \text{UNIV} - \{p\}. \text{hasRead} s p d q) \}\)

by (auto simp add: EndPhase2-def HInv5-inner-def)

\textbf{case True}

have \( p32: \text{maxBalInp} s (\text{bal}(\text{dblock} s p)) (\text{inp}(\text{dblock} s p)) \)

proof -

have \( \neg(\exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q) \land 
\neg \text{hasRead} s p d q)) \)

proof auto

fix \( D q \)

assume \( Dmaj: D \in \text{MajoritySet} \)

from \( \text{ep2-maj Dmaj majoritys-intersect} \)

have \( \exists d \in D. d \in \text{disksWritten} s p \land
(\forall q \in \text{UNIV} - \{p\}. \text{hasRead} s p d q) \)

by (auto simp add: MajoritySet-def, blast)

then obtain \( d \)

where dinD: \( d \in D \)

and ddisk: \( d \in \text{disksWritten} s p \)

and dhasR: \( \forall q \in \text{UNIV} - \{p\}. \text{hasRead} s p d q \)

by auto

from \( \text{inv2b} \)

have \( \text{Inv2b-inner} s p d \)

by (auto simp add: Inv2b-def)

with ddisk

have \( \text{disk} s d p = \text{dblock} s p \)

by (auto simp add: Inv2b-inner-def)

with \( \text{inv2c phase} \)

have \( \text{bal}(\text{dblock} s p) = \text{mbal}(\text{disk} s d q) \)

by (auto simp add: Inv2c-def Inv2c-inner-def)

with \( \text{dhasR} \in D \)

show \( \exists d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q) \rightarrow \text{hasRead} s p d q \)

by auto

qed

with \( \text{inv5R} \)

show \( ?\text{thesis} \)

by (auto simp add: HInv5-inner-R-def)

\textbf{qed}

have \( p33: \text{maxBalInp} s' (\text{bal}(\text{dblock} s' p)) (\text{chosen} s') \)

proof -

from \( \text{act} \)

have \( \text{outpt': outpt} s' = (\text{outpt} s) (p := \text{inp}(\text{dblock} s p)) \)

by (auto simp add: EndPhase2-def)

have \( \text{outpt' q: } \forall q. p \neq q \rightarrow \text{outpt} s' q = \text{NotAnInput} \)

proof auto

fix \( q \)

assume \( pnq: p \neq q \)

from \( \text{outpt' pnq} \)

have \( \text{outpt} s' q = \text{outpt} s q \)

by (auto simp add: EndPhase2-def)

with \( \text{True inv2c} \)
show \( \text{outpt } s' \ q = \text{NotAnInput} \)
by (auto simp add: Inv2c-def Inv2c-inner-def)
qed

from True act chosen'
have chosen s' = inp (dblock s p)
proof (auto simp add: HNextPart-def split: split-if_asm)
  fix pa
  assume outpt'-pa: outpt s' pa \neq \text{NotAnInput} 
  from outpt'-q
  have someeq2: \( \forall pa. \\text{outpt } s' pa \neq \text{NotAnInput} \implies pa=p \)
  by auto
  with outpt'-pa
  have outpt s' p \neq \text{NotAnInput} 
  by auto
  from some-equality[of \( \lambda p. \\text{outpt } s' p \neq \text{NotAnInput} \), OF this someeq2]
  have (SOME p. outpt s' p \neq \text{NotAnInput}) = p
  with outpt'
  show outpt s' (SOME p. outpt s' p \neq \text{NotAnInput}) = inp (dblock s p)
  by auto
qed

moreover 
from act 
have bal(dblock s' p) = bal(dblock s p)
by (auto simp add: EndPhase2-def)
ultimately 
have maxBalInp s (bal(dblock s' p)) (chosen s')
  using p32
  by auto
  with HEndPhase2-allBlocks[OF act]
show ?thesis 
  by (auto simp add: maxBalInp-def)
qed

from ep2-maj inv2b majorities-intersect 
have \( \exists D \in \text{MajoritySet}. \ (\forall d \in D. \ \text{disk } s \ d \ p = \text{dblock } s \ p \)
\and (\forall q \in \text{UNIV} - \{p\}. \text{hasRead } s \ p \ d \ q) \)
by (auto simp add: Inv2b-def Inv2b-inner-def MajoritySet-def)
then obtain D 
where Dmaj: D \in MajoritySet 
and p34: \( \forall d \in D. \ \text{disk } s \ d \ p = \text{dblock } s \ p \)
\and (\forall q \in \text{UNIV} - \{p\}. \text{hasRead } s \ p \ d \ q) 
by auto
have p35: \( \forall q. \forall d \in D. (\text{phase } s \ q = 1 \ \land \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s \ q) \land \text{hasRead } s \ q \ d) \)
    \rightarrow (\{\text{block} = \text{dblock } s \ p, \text{proc} = p\} \in \text{blocksRead } s \ q \ d)
proof auto
  fix q d
  assume dD: d \in D and phase-q: phase s q = Suc 0
  and bal-mbal: \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s \ q) \and \text{hasRead}: \text{hasRead } s \ q \ d \ p
from phase inv2c

136
have \( \text{bal}(\text{dblock } s \ p) = \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: Inv2c-def Inv2c-inner-def)

moreover
from inv2c phase
have \( \forall \text{br} \in \text{blocksRead } s \ p \ d. \text{mbal}(\text{block } \text{br}) < \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: Inv2c-def Inv2c-inner-def)

ultimately
have \( p_{41} : (| \text{block} = \text{dblock } s \ q, \text{proc} = q |) \notin \text{blocksRead } s \ p \ d \)
using bal-mbal
by auto

from phase phase-q
have \( p \neq q \) by auto
with \( p_{34} \ dD \)
have \( \text{hasRead } s \ p \ d \ q \)
by auto
with phase phase-q hasRead inv3 p_{41}
show \( (| \text{block} = \text{dblock } s \ p, \text{proc} = p |) \notin \text{blocksRead } s \ q \ d \)
by (auto simp add: HInv3-def HInv3-inner-def HInv3-L-def HInv3-R-def)
qed

have \( p_{36} : \forall q. \forall d \in D. \text{phase } s' = 1 \wedge \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s' \ q) \wedge \text{hasRead } s' \ q \ d \)
\( \rightarrow (\exists \text{br} \in \text{blocksRead } s' \ q \ d. \text{bal}(\text{block } \text{br}) = \text{bal}(\text{dblock } s \ p)) \)

proof (auto)
fix \( q \ d \)
assume \( dD : d \in D \) and phase-q: phase s' q = Suc 0
and bal: bal (dblock s p) \( \leq \text{mbal} \) (dblock s' q)
and hasRead: hasRead s' q d p
from phase-q act
have phase s' q = phase s q \& \text{dblock } s' q = \text{dblock } s q \& \text{hasRead } s' q d p = \text{hasRead } s q d p \wedge \text{blocksRead } s' q d = \text{blocksRead } s q d
by (auto simp add: EndPhase2-def hasRead-def InitializePhase-def)
with \( p_{35} \) phase-q bal hasRead dD
have \( (| \text{block} = \text{dblock } s \ p, \text{proc} = p |) \in \text{blocksRead } s' q d \)
by auto
thus \( \exists \text{br} \in \text{blocksRead } s' q d. \text{bal}(\text{block } \text{br}) = \text{bal}(\text{dblock } s \ p) \)
by force
qed

hence \( p_{36-2} : \forall q. \forall d \in D. \text{phase } s' = 1 \wedge \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s' \ q) \wedge \text{hasRead } s' q d p \rightarrow (\exists \text{br} \in \text{blocksRead } s' q d. \text{bal}(\text{block } \text{br}) \leq \text{bal}(\text{block } \text{br})) \)

by force
from act
have \( \text{bal-dblock}: \text{bal}(\text{dblock } s' p) = \text{bal}(\text{dblock } s \ p) \)
and disk: disk s' = disk s
by (auto simp add: EndPhase2-def)
from \( \text{bal-dblock } p_{33} \)
have \( \text{maxBalInp } s' (\text{bal}(\text{dblock } s \ p)) \) (chosen s')
by auto
moreover
from disk p34
have \( \forall d \in D. \, \text{bal}(\text{dblock} \, s \, p) \leq \text{bal}(\text{disk} \, s' \, d \, p) \)
  by auto
ultimately
have \( \text{maxBalInp} \, s' \, (\text{bal}(\text{dblock} \, s \, p)) \, (\text{chosen} \, s') \land \)
  (\( \exists D \in \text{MajoritySet} \).
  \( \forall d \in D. \, \text{bal}(\text{dblock} \, s \, p) \leq \text{bal}(\text{disk} \, s' \, d \, p) \land \)
  (\( \forall q. \, \text{phase} \, s' \, q = \text{Suc} \, 0 \land \)
  \( \text{bal}(\text{dblock} \, s \, p) \leq \text{mbal}(\text{dblock} \, s' \, q) \land \text{hasRead} \, s' \, q \, d \, p \rightarrow \)
  (\( \exists \text{br} \in \text{blocksRead} \, s' \, q \, d. \, \text{bal}(\text{dblock} \, s \, p) \leq \text{bal}(\text{block} \, \text{br}))\))
  using p36-2 Dmaj
  by auto
moreover
from phase inv2c
have \( \text{bal}(\text{dblock} \, s \, p) \in \text{Ballot} \, p \)
  by (auto simp add: Inv2c-def Inv2c-inner-def)
ultimately
show \( \text{thesis} \)
  by (auto simp add: valueChosen-def)
next
case False
with act
have p31: \( \text{chosen} \, s' = \text{chosen} \, s \)
  by (auto simp add: HNextPart-def)
from False inv
have valueChosen s (chosen s)
  by (auto simp add: HInv6-def)
from HEndPhase2-valueChosen[OF act this] p31 False InputsOrNi
show \( \text{thesis} \)
  by auto
qed

lemma valueChosen-equal-case:
  assumes max-v: \( \text{maxBalInp} \, s \, b \, v \)
  and Dmaj: \( D \in \text{MajoritySet} \)
  and asm-v: \( \forall d \in D. \, b \leq \text{bal}(\text{disk} \, s \, d \, p) \)
  and max-w: \( \text{maxBalInp} \, s \, b \, w \)
  and Damaj: \( Da \in \text{MajoritySet} \)
  and asm-w: \( \forall d \in Da. \, b \leq \text{bal}(\text{disk} \, s \, d \, pa) \)
  and b-ba: \( b \leq ba \)
  shows \( v = w \)
proof –
  have \( \forall d. \, \text{disk} \, s \, d \, pa \in \text{allBlocks} \, s \)
    by (auto simp add: allBlocks-def blocksOf-def)
  with majorities-intersect Dmaj Damaj
  have \( \exists d \in D \cap Da. \, \text{disk} \, s \, d \, pa \in \text{allBlocks} \, s \)
    by (auto simp add: MajoritySet-def, blast)
  then obtain d
where \( \text{dInmaj} \): \( d \in D \cap Da \) and \( \text{dab} \): disk \( s d p \in \text{allBlocks} s \)
by auto
with \( \text{asm-w} \)
have \( \text{ba} \): \( \text{ba} \leq \text{bal} \) (disk \( s d p \))
by auto
with \( \text{b-ba} \)
have \( \text{b} \leq \text{bal} \) (disk \( s d p \))
by auto
with \( \text{max-v dab} \)
have \( \text{v-value} \): \( \text{inp} \) (disk \( s d p \)) = \( v \)
by (auto simp add: \( \text{maxBalInp-def} \))
from \( \text{ba max-w dab} \)
have \( \text{w-value} \): \( \text{inp} \) (disk \( s d p \)) = \( w \)
by (auto simp add: \( \text{maxBalInp-def} \))
with \( \text{v-value} \)
show \( \text{\textit{thesis}} \) by auto
qed

lemma \( \text{valueChosen-equal} \):
assumes \( v \): \( \text{valueChosen s v} \)
and \( w \): \( \text{valueChosen s w} \)
shows \( v = w \) using \( \text{assms} \)
proof (auto simp add: \( \text{valueChosen-def} \))
fix \( a b a a b a p D p a D a \)
assume \( \text{max-v: maxBalInp s b v} \)
and \( \text{Dmaj: D \in MajoritySet} \)
and \( \text{asm-v: \( \forall \) d \in D. \text{b} \leq \text{bal} \) (disk \( s d p \)) \land \)
\( (\forall q. \text{phase s q} = \text{Suc 0} \land \)
\( \text{b} \leq \text{mbal} \) (dblock s q) \land \text{hasRead s q d p} \rightarrow \)
\( (\exists \text{br} \in \text{blocksRead s q d}. \text{b} \leq \text{bal} \) (block \text{br})))
and \( \text{max-w: maxBalInp s ba w} \)
and \( \text{Dmaj: Da \in MajoritySet} \)
and \( \text{asm-w: \( \forall \) d \in Da. \text{ba} \leq \text{bal} \) (disk \( s d p \)) \land \)
\( (\forall q. \text{phase s q} = \text{Suc 0} \land \)
\( \text{ba} \leq \text{mbal} \) (dblock s q) \land \text{hasRead s q d p} \rightarrow \)
\( (\exists \text{br} \in \text{blocksRead s q d}. \text{ba} \leq \text{bal} \) (block \text{br})))
from \( \text{asm-v} \)
have \( \text{asm-v: \( \forall \) d \in D. \text{b} \leq \text{bal} \) (disk \( s d p \)) \) by auto
from \( \text{asm-w} \)
have \( \text{asm-w: \( \forall \) d \in Da. \text{ba} \leq \text{bal} \) (disk \( s d p \)) \) by auto
show \( v = w \)
proof (cases \( b \leq \text{ba} \))
case True
from \( \text{valueChosen-equal-case[OF max-v Dmaj asm-v max-w Dmaj asm-w True]} \)
show \( \text{\textit{thesis}} \).
next
case False
from \( \text{valueChosen-equal-case[OF max-w Dmaj asm-w max-v Dmaj asm-v]} \)
False
show ?thesis
  by auto
qed

lemma HEndPhase2-Inv6-2:
assumes act: HEndPhase2 s s' p
and inv: HInv6 s
and inv2b: Inv2b s
and inv2c: Inv2c s
and inv3: HInv3 s
and inv5: HInv5-inner s p
and asm: outpt s' r ≠ NotAnInput
shows outpt s' r = chosen s'
proof (cases chosen s=NotAnInput)
  case True
  with inv2c
  have ∃ q. outpt s q = NotAnInput
    by (auto simp add: Inv2c-def Inv2c-inner-def)
  with True act asm
  show ?thesis
    by (auto simp add: EndPhase2-def HNextPart-def
      split: split-if-asm)
next
  case False
  with inv
  have p31: valueChosen s (chosen s)
    by (auto simp add: HInv6-def)
  with False act
  have chosen s'≠ NotAnInput
    by (auto simp add: HNextPart-def)
  from HEndPhase2-Inv6-1[OF act inv inv2b inv2c inv3 inv5 this]
  have p32: valueChosen s'(chosen s').
  from False InputsOrNi
  have chosen s ∈ Inputs by auto
  from valueChosen-equal[OF HEndPhase2-valueChosen[OF act p31 this] p32]
  have p33: chosen s = chosen s'.
  from act
  have maj: IsMajority {d. d ∈ disksWritten s p
    ∧ (∀ q ∈ UNIV − {p}. hasRead s p d q)} (is IsMajority ?D)
    and phase: phase s p = 2
  by (auto simp add: EndPhase2-def)
  show ?thesis
proof (cases outpt s r = NotAnInput)
  case True
  with asm act
  have p41: r=p
    by (auto simp add: EndPhase2-def split: split-if-asm)
  from maj
have \( p42: \exists D \in \text{MajoritySet}. \forall d \in D. \forall q \in \text{UNIV} - \{p\}. \text{hasRead} s p d q \)
by (auto simp add: MajoritySet-def)

have \( p43: \neg(\exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \quad \text{bal}(\text{dblock} s p d q) < \text{mbal}(\text{disk} s d q) \quad \land \neg \text{hasRead} s p d q)) \)
proof auto
fix \( D q \)
assume \( Dmaj: D \in \text{MajoritySet} \)
show \( \exists d \in D. \text{bal}(\text{dblock} s p d) < \text{mbal}(\text{disk} s d q) \quad \land \neg \text{hasRead} s p d q \)
proof (cases \( p=q \))
assume \( pq: p=q \)
thus ?thesis
proof auto
from \( \text{maj} \) \( \text{majorities-intersect Dmaj} \)
have \( ?D \cap D \neq \{\} \)
by (auto simp add: MajoritySet-def)
hence \( \exists d \in ?D \cap D. d \in \text{disksWritten} s p \) by auto
then obtain \( d \) where \( d \in \text{disksWritten} s p \) and \( d \in ?D \cap D \)
by auto
hence \( dD: d \in D \) by auto
from \( d \) \( \text{inv2b} \)
have \( \text{disk} s d p = \text{dblock} s p \)
by (auto simp add: Inv2b-def Inv2b-inner-def)
with \( \text{inv2c} \) \( \text{phase} \)
have \( \text{bal}(\text{dblock} s p) = \text{mbal}(\text{disk} s d p) \)
by (auto simp add: Inv2c-def Inv2c-inner-def)
with \( dD \) \( pq \)
show \( \exists d \in D. \text{bal}(\text{dblock} s q) < \text{mbal}(\text{disk} s d q) \quad \land \neg \text{hasRead} s q d q \)
by auto
qed
next
  case False
with \( p42 \)
have \( \exists D \in \text{MajoritySet}. \forall d \in D. \text{hasRead} s p d q \)
by auto
with \( \text{majorities-intersect Dmaj} \)
show ?thesis
by (auto simp add: MajoritySet-def, blast)
qed

have \( p44: \text{maxBalInp} s (\text{bal}(\text{dblock} s p)) (\text{inp}(\text{dblock} s p)) \)
by (auto simp add: EndPhase2-def HInv5-inner-def HInv5-inner-R-def)

have \( \exists bk \in \text{allBlocks} s. \exists b \in (\text{UN} p. \text{Ballot} p). (\text{maxBalInp} s b (\text{chosen} s)) \land b \leq \text{bal} bk \)
proof -
have \( \text{disk-allblks}: \forall d \in \text{allBlocks} s. \text{disk} s d p \in \text{allBlocks} s \)
by (auto simp add: allBlocks-def blocksOf-def)

141
from p31
have \( \exists b \in (\bigcup p. \text{Ballot } p), \max \text{BalInp } s \ b \ (\text{chosen } s) \land \\
(\exists p, \exists D \in \text{MajoritySet}. (\forall d \in D. \ b \leq bal(disk s d p))) \)
  by (auto simp add: valueChosen-def, force)
with majority-nonempty obtain b p D d
  where IsMajority \( D \land b \in (\bigcup p. \text{Ballot } p) \land \\
\max \text{BalInp } s \ b \ (\text{chosen } s) \land d \in D \land b \leq bal(disk s d p) \)
  by (auto simp add: MajoritySet-def, blast)
with disk-allblks
show ?thesis
  by (auto)
qed
then obtain bk b
  where p45-bk: \( bk \in \text{allBlocks } s \land b \leq bal \ bk \)
  and p45-b: \( b \in (\bigcup p. \text{Ballot } p) \land (\max \text{BalInp } s \ b \ (\text{chosen } s)) \)
  by auto
have p46: \( \text{inp(dblock } s \ p) = \text{chosen } s \)
proof (cases \( b \leq bal(dblock s p) \))
  case True
  have dblock s p \in \text{allBlocks } s
    by (auto simp add: allBlocks-def blocksOf-def)
  with p45-b True
  show ?thesis
    by (auto simp add: maxBalInp-def)
next
  case False
  from p44 p45-bk False
  have \( \text{inp } bk = \text{inp(dblock } s \ p) \)
    by (auto simp add: maxBalInp-def)
  with p45-b p45-bk
  show ?thesis
    by (auto simp add: maxBalInp-def)
qed
with p41 p33 act
show ?thesis
  by (auto simp add: EndPhase2-def)
next
  case False
  from inv2c
  have Inv2c-inner s r
    by (auto simp add: Inv2c-def)
  with False assm inv2c act
  have \( \text{outpt } s' = \text{outpt } s \ r \)
    by (auto simp add: Inv2c-inner-def EndPhase2-def
                      split: split-if-asm)
  with inv p33 False
  show ?thesis
    by (auto simp add: HInv6-def)
qed
qed

theorem HEndPhase2-Inv6:
  assumes act: HEndPhase2 s s' p
  and inv: HInv6 s
  and inv2b: Inv2b s
  and inv2c: Inv2c s
  and inv3: HInv3 s
  and inv5: HInv5-inner s p
  shows HInv6 s'
proof (auto simp add: HInv6-def)
assume chosen s' \neq NotAnInput
from HEndPhase2-Inv6-1[OF act inv inv2b inv2c inv3 inv5 this]
show valueChosen s' (chosen s') .
next
fix p
assume outpt s' p \neq NotAnInput
from HEndPhase2-Inv6-2[OF act inv inv2b inv2c inv3 inv5 this]
show outpt s' p = chosen s' .
qed

lemma outpt-chosen:
  assumes outpt: outpt s = outpt s'
  and inv2c: Inv2c s
  and nextp: HNextPart s s'
  shows chosen s' = chosen s
proof
  from inv2c
  have chosen s = NotAnInput \rightarrow (\forall p. outpt s p = NotAnInput)
    by (auto simp add: Inv2c-inner-def Inv2c-def)
  with outpt nextp
  show ?thesis
    by (auto simp add: HNextPart-def)
qed

lemma outpt-Inv6:
[ outpt s = outpt s'; \forall p. outpt s p \in \{chosen s, NotAnInput\} ;
  Inv2c s ; HNextPart s s' ] \implies \forall p. outpt s' p \in \{chosen s', NotAnInput\}
using assms and outpt-chosen
by auto

theorem HStartBallot-Inv6:
  assumes act: HStartBallot s s' p
  and inv: HInv6 s
  and inv2c: Inv2c s
  shows HInv6 s'
proof
  from outpt-chosen act inv2c inv
  have chosen s' \neq NotAnInput \rightarrow valueChosen s (chosen s')
by(auto simp add: StartBallot-def HInv6-def)
from HStartBallot-valueChosen[OF act] this InputsOrNi
have t1: chosen s' ≠ NotAnInput → valueChosen s' (chosen s')
  by auto
from act
have outpt: outpt s = outpt s'
  by(auto simp add: StartBallot-def)
from outpt-Inv6[OF outpt] act inv2c inv
have ∀p. outpt s' p = chosen s' ∨ outpt s' p = NotAnInput
  by(auto simp add: HInv6-def)
with t1
show ?thesis
  by(simp add: HInv6-def)
qed

theorem HPhase1or2Write-Inv6:
  assumes act: HPhase1or2Write s s' p d
  and inv: HInv6 s
  and inv4: HInv4a s p
  and inv2c: Inv2c s
  shows HInv6 s'
proof –
  from outpt-chosen act inv2c inv
  have chosen s' ≠ NotAnInput → valueChosen s (chosen s')
    by(auto simp add: Phase1or2Write-def HInv6-def)
  from HPhase1or2Write-valueChosen[OF act] inv4 this InputsOrNi
  have t1: chosen s' ≠ NotAnInput → valueChosen s' (chosen s')
    by auto
  from act
  have outpt: outpt s = outpt s'
    by(auto simp add: Phase1or2Write-def)
  from outpt-Inv6[OF outpt] act inv2c inv
  have ∀p. outpt s' p = chosen s' ∨ outpt s' p = NotAnInput
    by(auto simp add: HInv6-def)
  with t1
  show ?thesis
    by(simp add: HInv6-def)
qed

theorem HPhase1or2ReadThen-Inv6:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv6 s
  and inv2c: Inv2c s
  shows HInv6 s'
proof –
  from outpt-chosen act inv2c inv
  have chosen s' ≠ NotAnInput → valueChosen s (chosen s')
    by(auto simp add: Phase1or2ReadThen-def HInv6-def)
  from HPhase1or2ReadThen-valueChosen[OF act] this InputsOrNi
have $t_1$: chosen $s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s' (\text{chosen } s')$

by auto

from act

have outpt: outpt $s = \text{outpt } s'$
  by (auto simp add: Phase1or2ReadThen-def)

from outpt-Inv6[OF outpt] act inv2c inv

have $\forall p. \text{outpt } s' p = \text{chosen } s' \lor \text{outpt } s' p = \text{NotAnInput}$
  by (auto simp add: HInv6-def)

with $t_1$

show ?thesis
  by (simp add: HInv6-def)

qed

**Theorem HPhase1or2ReadElse-Inv6**:

assumes act: $\text{HPhase1or2ReadElse } s \ s' \ p \ d \ q$

and inv: $\text{HInv6 } s$

and inv2c: $\text{Inv2c } s$

shows $\text{HInv6 } s'$

using assms and HStartBallot-Inv6

proof

from outpt-chosen act inv2c inv

have chosen $s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s' (\text{chosen } s')$
  by (auto simp add: EndPhase1-def HInv6-def)

from HEndPhase1-valueChosen[OF act] inv1 inv2a inv2b this InputsOrNi

have $t_1$: chosen $s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s' (\text{chosen } s')$
  by auto

from act

have outpt: outpt $s = \text{outpt } s'$
  by (auto simp add: EndPhase1-def)

from outpt-Inv6[OF outpt] act inv2c inv

have $\forall p. \text{outpt } s' p = \text{chosen } s' \lor \text{outpt } s' p = \text{NotAnInput}$
  by (auto simp add: HInv6-def)

with $t_1$

show ?thesis
  by (simp add: HInv6-def)

qed

**Lemma outpt-chosen-2**:

assumes outpt: $\text{outpt } s' = (\text{outpt } s) (p:= \text{NotAnInput})$
and \text{inv2c}: \text{Inv2c } s \\
and \text{nextp}: \text{HNextPart } s \ s' \\
shows\ chosen\ s = chosen\ s'
proof – 
from \text{inv2c} 
have chosen\ s = \text{NotAnInput} \rightarrow (\forall p.\ \text{outpt } s\ p = \text{NotAnInput}) 
by(auto simp add: \text{Inv2c-inner-def Inv2c-def}) 
with \text{outpt nextp} 
show ?thesis 
by(auto simp add: \text{HNextPart-def})
qed

lemma \text{outpt-HInv6-2}: 
assumes \text{outpt}: \text{outpt } s' = (\text{outpt } s)\ (p := \text{NotAnInput}) 
and \text{inv}: \forall p.\ \text{outpt } s\ p \in \{\text{chosen } s,\ \text{NotAnInput}\} 
and \text{inv2c}: \text{Inv2c } s \\
and \text{nextp}: \text{HNextPart } s \ s' \\
shows\ \forall p.\ \text{outpt } s'\ p \in \{\text{chosen } s',\ \text{NotAnInput}\} 
proof – 
from \text{outpt-chosen-2}[OF \text{outpt inv2c nextp}] 
have chosen\ s = chosen\ s'. 
with \text{inv outpt} 
show ?thesis 
by auto
qed

theorem \text{HFail-Inv6}: 
assumes \text{act}: \text{HFail } s \ s'\ p 
and \text{inv}: \text{HInv6 } s 
and \text{inv2c}: \text{Inv2c } s \\
shows\ \text{HInv6 } s' 
proof – 
from \text{outpt-chosen-2 \ act \ inv2c \ inv} 
have chosen\ s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s\ (\text{chosen } s') 
by(auto simp add: \text{Fail-def HInv6-def}) 
from \text{HFail-valueChosen}[OF \text{act}]\ this\ \text{InputsOrNi} 
have t1: chosen\ s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s'\ (\text{chosen } s') 
by auto 
from \text{act} 
have \text{outpt}: \text{outpt } s' = (\text{outpt } s)\ (p := \text{NotAnInput}) 
by(auto simp add: \text{Fail-def}) 
from \text{outpt-HInv6-2}[OF \text{outpt}]\ \text{act \ inv2c \ inv} 
have \forall p.\ \text{outpt } s'\ p = \text{chosen } s'\ \vee \text{outpt } s'\ p = \text{NotAnInput} 
by(auto simp add: \text{HInv6-def}) 
with t1 
show ?thesis 
by(simp add: \text{HInv6-def})
qed
\begin{verbatim}
theorem HPhase0Read-Inv6:
  assumes act: HPhase0Read s s' p d
  and inv: HInv6 s
  and inv2c: Inv2c s
  shows HInv6 s'
proof -
  from outpt-chosen act inv2c inv
  have chosen s' \neq NotAnInput \implies valueChosen s (chosen s')
    by (auto simp add: Phase0Read-def HInv6-def)
  from HPhase0Read-valueChosen[OF act] this InputsOrNi
  have t1: chosen s' \neq NotAnInput \implies valueChosen s'(chosen s')
    by auto
  from act
  have outpt: outpt s = outpt s'
    by (auto simp add: Phase0Read-def)
  from outpt-Inv6[OF outpt] act inv2c inv
  have \forall p. outpt s' p = chosen s' \vee outpt s' p = NotAnInput
    by (auto simp add: HInv6-def)
  with t1
  show ?thesis
    by (simp add: HInv6-def)
qed

theorem HEndPhase0-Inv6:
  assumes act: HEndPhase0 s s' p
  and inv: HInv6 s
  and inv1: Inv1 s
  and inv2c: Inv2c s
  shows HInv6 s'
proof -
  from outpt-chosen act inv2c inv
  have chosen s' \neq NotAnInput \implies valueChosen s (chosen s')
    by (auto simp add: EndPhase0-def HInv6-def)
  from HEndPhase0-valueChosen[OF act] inv1 this InputsOrNi
  have t1: chosen s' \neq NotAnInput \implies valueChosen s'(chosen s')
    by auto
  from act
  have outpt: outpt s = outpt s'
    by (auto simp add: EndPhase0-def)
  from outpt-Inv6[OF outpt] act inv2c inv
  have \forall p. outpt s' p = chosen s' \vee outpt s' p = NotAnInput
    by (auto simp add: HInv6-def)
  with t1
  show ?thesis
    by (simp add: HInv6-def)
qed

HInv1 \land HInv2 \land HInv2' \land HInv3 \land HInv4 \land HInv5 \land HInv6 is an invariant
of HNext.
\end{verbatim}
lemma I2f:
assumes nxt: HNext s s'
and inv: HInv1 s ∧ HInv2 s ∧ HInv2 s' ∧ HInv3 s ∧ HInv4 s ∧ HInv5 s ∧ HInv6 s
shows HInv6 s' using assms
by (auto simp add: HNext-def Next-def, 
  auto simp add: HInv2-def intro: HStartBallot-Inv6, 
  auto simp add: HInv4-def intro: HPhase1or2Write-Inv6, 
  auto simp add: Phase1or2Read-def 
    intro: HPhase1or2ReadThen-Inv6 
    HPhase1or2ReadElse-Inv6, 
  auto simp add: EndPhase1or2-def HInv1-def HInv5-def 
    intro: HEndPhase1-Inv6 
    HEndPhase2-Inv6, 
  auto intro: HFail-Inv6, 
  auto intro: HEndPhase0-Inv6)
end

theory DiskPaxos-Invariant imports DiskPaxos-Inv6 begin

C.8 The Complete Invariant

definition HInv :: state ⇒ bool
where
HInv s = (HInv1 s 
∧ HInv2 s 
∧ HInv3 s 
∧ HInv4 s 
∧ HInv5 s 
∧ HInv6 s)

theorem I1:
HInit s ⇒ HInv s 
using HInit-HInv1 HInit-HInv2 HInit-HInv3 
  HInit-HInv4 HInit-HInv5 HInit-HInv6 
by (auto simp add: HInv-def)

theorem I2:
assumes inv: HInv s
and nxt: HNext s s'
shows HInv s' 
by (simp add: HInv-def)
theory DiskPaxos imports DiskPaxos-Invariant begin

C.9 Inner Module

record Istate =
  iinput :: Proc ⇒ InputsOrNi
  ioutput :: Proc ⇒ InputsOrNi
  ichosen :: InputsOrNi
  iallInput :: InputsOrNi set

definition IInit :: Istate ⇒ bool
where
  IInit s = (range (iinput s) ⊆ Inputs
            ∧ ioutput s = (λp. NotAnInput)
            ∧ ichosen s = NotAnInput
            ∧ iallInput s = range (iinput s))

definition IChoose :: Istate ⇒ Istate ⇒ Proc ⇒ bool
where
  IChoose s s' p = (ioutput s p = NotAnInput
                   ∧ (if (ichosen s = NotAnInput)
                       then (∃ip ∈ iallInput s. ichosen s' = ip
                             ∧ ioutput s' = (ioutput s) (p := ip))
                       else ( ioutput s' = (ioutput s) (p:= ichosen s)
                             ∧ ichosen s' = ichosen s))
                   ∧ iinput s' = iinput s ∧ iallInput s' = iallInput s)

definition IFail :: Istate ⇒ Istate ⇒ Proc ⇒ bool
where
  IFail s s' p = (ioutput s' = (ioutput s) (p:= NotAnInput)
                 ∧ (∃ip ∈ Inputs. iinput s' = (iinput s)(p:= ip)
                   ∧ iallInput s' = iallInput s ∪ {ip})
                 ∧ ichosen s' = ichosen s)

definition INext :: Istate ⇒ Istate ⇒ bool
where
  INext s s' = (∃ p. IChoose s s' p ∨ IFail s s' p)

definition s2is :: state ⇒ Istate
where
  s2is s = (iinput = inpt s,
            ioutput = outpt s,
            ichosen=chosen s,
            iallInput = allInput s)

theorem R1:
\[ \text{HInit } s; \text{ is } = s2is \text{ s} \]  \implies \text{HInit is} \\
\text{by (auto simp add: HInit-def HInit-def s2is-def Init-def)}

\text{theorem R2b:} \\
\text{assumes inv: HInv s} \\
\text{and inv': HInv s'} \\
\text{and nxt: HNext s s'} \\
\text{and srel: is=s2is s \land is'=s2is s'} \\
\text{shows } (\exists p. \text{IFail is is'} p \lor \text{IChoose is is'} p) \lor \text{is } = \text{is'} \\
\text{proof (auto)} \\
\text{assume chg-vars: is\neq is'} \\
\text{with srel} \\
\text{have s-change: inpt s \neq inpt s' \lor outpt s \neq outpt s'} \\
\text{\lor chosen s \neq chosen s' \lor allInput s \neq allInput s'} \\
\text{by (auto simp add: s2is-def)} \\
\text{from inv} \\
\text{have inv2c5: } \forall p. \text{ inpt s p } \in \text{allInput s} \\
\text{\land (chosen s } = \text{ NotAnInput } \implies \text{ outpt s p } = \text{ NotAnInput} ) \\
\text{by (auto simp add: HInv-def HInv2-def Inv2c-def Inv2c-inner-def)} \\
\text{from nxt s-change inv2c5} \\
\text{have inpt s' \neq inpt s \lor outpt s' \neq outpt s} \\
\text{by (auto simp add: HNext-def Next-def HNextPart-def)} \\
\text{with nxt} \\
\text{have } (\exists p. \text{Fail s s'} p \lor \text{EndPhase2 s s'} p) \\
\text{by (auto simp add: HNext-def Next-def StartBallot-def Phase0Read-def Phase1or2Write-def} \\
\text{Phase1or2Read-def Phase1or2ReadThen-def Phase1or2ReadElse-def} \\
\text{EndPhase1or2-def EndPhase1-def EndPhase0-def)} \\
\text{then obtain } p \text{ where fail-or-endphase2: Fail s s' p \lor EndPhase2 s s' p} \\
\text{by auto} \\
\text{from inv} \\
\text{have inv2c: Inv2c-inner s p} \\
\text{by (auto simp add: HInv-def HInv2-def Inv2c-def)} \\
\text{from fail-or-endphase2 have IFail is is' p \lor IChoose is is' p} \\
\text{proof} \\
\text{assume fail: Fail s s' p} \\
\text{hence phase': phase s' p = 0} \\
\text{and outpt: outpt s' } = (\text{outpt s}) (p:= \text{ NotAnInput}) \\
\text{by (auto simp add: Fail-def)} \\
\text{have IFail is is' p} \\
\text{proof} \\
\text{from fail srel} \\
\text{have ioutput is' } = (\text{ioutput is}) (p:= \text{ NotAnInput}) \\
\text{by (auto simp add: Fail-def s2is-def)} \\
\text{moreover} \\
\text{from nxt} \\
\text{have all-nxt: allInput s' } = \text{ allInput s } \cup (\text{range (inpt s')}) \\
\text{by (auto simp add: HNext-def HNextPart-def)} \\
\text{from fail srel}
have \( \exists ip \in Inputs. \ iinput is' = (iinput is)(p:= ip) \)
  by (auto simp add: Fail-def s2is-def)
then obtain ip where ip-Input: \( ip \in Inputs \) and
  \( iinput is' = (iinput is)(p:= ip) \)
  by auto
with inv2c5 srel all-nxt
have ichosen is' = ichosen is
  by (auto simp add: HNext-def HNextPart-def s2is-def Inv2c-inner-def)
ultimately
show ?thesis
  using ip-Input
  by (auto simp add: IFail-def)
qed
thus ?thesis
  by auto
next
assume endphase2: EndPhase2 s s' p
from endphase2
have phase s p = 2
  by (auto simp add: EndPhase2-def)
with inv2c Ballot-nzero
have bal-dblik-nzero: bal(dblock s p) \( \neq 0 \)
  by (auto simp add: Inv2c-inner-def)
moreover
from inv
have inv2a-dblik: Inv2a-innermost s p (dblock s p)
  by (auto simp add: Hinv-def Hinv2-def Inv2a-def Inv2a-inner-def blocksOf-def)
ultimately
have p22: inp (dblock s p) \( \in allInput s \)
  by (auto simp add: Inv2a-innermost-def)
from inv
have allInput s \( \subseteq Inputs \)
  by (auto simp add: Hinv-def Hinv1-def)
with p22 NotAnInput endphase2
have outpt-uni: outpt s' p \( \neq \) NotAnInput
  by (auto simp add: EndPhase2-def)
show ?thesis
proof(cases chosen s = NotAnInput)
  case True
  with inv2c5
  have p31: \( \forall q. \) outpt s q = NotAnInput
    by auto
  with endphase2
  have p32: \( \forall q \in UNIV - \{p\}. \) outpt s' q = NotAnInput
by (auto simp add: EndPhase2-def)
hence some-eq: (∀x. outpt s' x ≠ NotAnInput → x = p)
  by auto
from p32 True nxt some-equality[of λp. outpt s' p ≠ NotAnInput, OF outpt-nni
  some-eq]
  have p33: chosen s' = outpt s' p
    by (auto simp add: HNext-def HNextPart-def)
  with endphase2
  have chosen s' = inp(dblock s p) ∧ outpt s' = (outpt s)(p := inp(dblock s p))
    by (auto simp add: EndPhase2-def)
  with True p22
  have if (chosen s = NotAnInput)
    then (∃ip ∈ allInput s. chosen s' = ip
      ∧ outpt s' = (outpt s)(p := ip))
    else ( outpt s' = (outpt s)(p := chosen s)
      ∧ chosen s' = chosen s)
      by auto
moreover
from endphase2 inv2c5 nxt
have inp s' = inp s ∧ allInput s' = allInput s
  by (auto simp add: EndPhase2-def HNext-def HNextPart-def)
ultimately
show ?thesis
  using srel p31
  by (auto simp add: IChoose-def s2is-def)
next
case False
with nxt
have p31: chosen s' = chosen s
  by (auto simp add: HNext-def HNextPart-def)
from inv'
have inv6: HInv6 s'
  by (auto simp add: HInv-def)
have p32: outpt s' p = chosen s
proof−
  from endphase2
  have outpt s' p = inp(dblock s p)
    by (auto simp add: EndPhase2-def)
moreover
from inv6 p31
have outpt s' p ∈ {chosen s, NotAnInput}
  by (auto simp add: HInv6-def)
ultimately
show ?thesis
  using outpt-nni
  by auto
qed
from srel False
have IChoose is is' p

152
proof(clarsimp simp add: IChoose-def s2is-def)
  from endphase2 inv2c
  have outpt s p = NotAnInput
    by(auto simp add: EndPhase2-def Inv2c-inner-def)
  moreover
  from endphase2 p31 p32 False
  have outpt s' = (outpt s) (p:= chosen s) ∧ chosen s' = chosen s
    by(auto simp add: EndPhase2-def)
  moreover
  from endphase2 nxt inv2c5
  have inpt s' = inpt s ∧ allInput s'= allInput s
    by(auto simp add: EndPhase2-def HNext-def HNextPart-def)
  ultimately
  show outpt s p = NotAnInput
    ∧ outpt s' = (outpt s)(p := chosen s) ∧ chosen s' = chosen s
    ∧ inpt s' = inpt s ∧ allInput s'= allInput s
    by auto
  qed
  thus ?thesis
    by auto
  qed
  qed
  thus ∃p. IFail is is' p ∨ IChoose is is' p
    by auto
  qed
end