Simpler Proofs for Concurrent Reading and Writing

Fred B. Schneider

Abstract
Simplified proofs are given for Lamport’s protocols to coordinate concurrent reading and writing.

1 Introduction
In most computing systems, hardware ensures that read and write operations to some basic unit of memory can be considered mutually exclusive. As a result, a read that overlaps with a write is serialized and will appear either to precede that write or to follow it. Operations that make multiple accesses to memory are not serialized by the hardware. The programmer must ensure that when such operations overlap, they produce meaningful results.

In this paper, we give simplified proofs for two protocols proposed by Lamport [1] for coordinating read and write operations that involve multiple accesses to memory. The two key theorems in [1] are long and intricate. Here, we show that both are corollaries of a single, relatively simple theorem. Our facility with proofs and the use of formalism has improved significantly in a little over 15 years. This is due, in part, to the influence of Edsger Dijkstra.

2 Words from digits
Consider a computing system that contains a finite set of values that can be encoded in such a sequence of digits. A word is a sequence of digits. A word will not be serialized by the hardware. The programmer must ensure that when such operations overlap, they produce meaningful results.

In this paper, we give simplified proofs for two protocols proposed by Lamport [1] for coordinating read and write operations that involve multiple accesses to memory. The two key theorems in [1] are long and intricate. Here, we show that both are corollaries of a single, relatively simple theorem. Our facility with proofs and the use of formalism has improved significantly in a little over 15 years. This is due, in part, to the influence of Edsger Dijkstra.

2 Words from digits
Consider a computing system that contains a finite set of values that can be encoded in such a sequence of digits. A word is a sequence of digits. A word will not be serialized by the hardware. The programmer must ensure that when such operations overlap, they produce meaningful results.

In this paper, we give simplified proofs for two protocols proposed by Lamport [1] for coordinating read and write operations that involve multiple accesses to memory. The two key theorems in [1] are long and intricate. Here, we show that both are corollaries of a single, relatively simple theorem. Our facility with proofs and the use of formalism has improved significantly in a little over 15 years. This is due, in part, to the influence of Edsger Dijkstra.
2 Words from digits

Consider a computing system in which the basic unit of memory is a digit, and a digit can contain one of \( B \geq 2 \) distinct values. Any element from a finite set of values can be encoded using a finite sequence of digits. We call such a sequence of digits a word. To read the value stored by a word, read operations are performed on its digits; to write a value, write operations are performed. Observe that overlapping read and write operations to a word will not be serialized by the hardware. Therefore, without additional constraints on execution, it is possible for a read that overwrites a write to obtain a meaningless value. For example, suppose digits can encode integers from 0 through 9 and a word \( w \) constructed from three digits initially encodes the value 099. A read that is concurrent with a write of value 100 might obtain any of the following results: 099, 090, 009, 000, 199, 190, 109, 100.

By constraining the order in which digits are read and the order in which digits are written, we can ensure that a read overlapping one or more writes does obtain a meaningful value. Desired are constraints that are both easily implemented and non-intrusive. Execution of neither read nor write operations should be delayed, nor should the constraints require elaborate synchronization primitives.

In the protocols that follow, a word \( w \) is implemented by a sequence \( w[0]w[1]\ldots w[n] \) of digits. Think of \( w[0] \) as the most-significant (left-most) digit and \( w[n] \) as the least-significant (right-most) digit of a base \( B \) number being stored by \( w \). We assume that \( w \) is written by a single, sequential process. Define \( w[i]^p \) to be the value written to digit \( w[i] \) by write operation number \( p \). Also, for any sequence \( s = s[0]s[1]\ldots s[n] \), define \( s[i\ldots j] \) to be the subsequence consisting of \( s[i]\ldots s[j] \), and define \( |s| \) to be the length of \( s \). Thus, \( w[0\ldots k] \) is the word constructed from the most-significant (left-most) \( k+1 \) digits of \( w \).

A read operation that overlaps with one or more writes can obtain a value that corresponds to the result of no write operation. We can describe such values by using a slice, a sequence of positive integers. For a word \( w \) and a slice \( \sigma \) of equal length, define:

\[
w^\sigma = w[0]^{\sigma[0]} w[1]^{\sigma[1]} \ldots w[n]^{\sigma[n]}\]

We write \((N) \otimes v\) to denote a length \( N \) sequence of \( v \)'s. Thus, \( w^{(n+1) \otimes p} \) equals \( w[0]^p w[1]^p \ldots w[n]^p \), the value written to \( w \) by write operation number \( p \).

A slice \( \sigma \) is non-decreasing if \((\forall i : 0 < i < |\sigma| : \sigma[i-1] \leq \sigma[i])\) and

\(^3\)It will be convenient to assume that a write operation to a word writes a value to every digit. The new value can, of course, be the same as the old.
non-increasing if \((\forall i : 0 < i < |\sigma| : \sigma[i - 1] \geq \sigma[i])\). For slices \(\sigma\) and \(\tau\) such that \(|\sigma| = |\tau|\), define
\[
\sigma \subseteq \tau \iff (\forall i : 0 \leq i < |\sigma| : \sigma[i] \leq \tau[i]).
\]

Finally, in order to reason about the relative order in which operations occur, define \(\mu_i(x)\) to be the number of writes that have been made to digit \(w[i]\) as of time \(x\). Observe that \(x \leq x'\) implies that \(\mu_i(x) \leq \mu_i(x')\) is valid.

3 The main result

We first show that if slices \(\sigma\) and \(\tau\) satisfy certain restrictions (H1–H3) and values written to \(w\) are non-decreasing (H4), then \(w^\sigma \leq w^\tau\) where "\(\leq\)" denotes lexicographic ordering.

**Theorem 1** Let \(\sigma\) and \(\tau\) be slices such that \(|\sigma| = |\tau| \geq N + 1\). Then,
\[
w[0..N]^\sigma \leq w[0..N]^\tau
\]
provided:

(H1) \(\sigma\) is non-decreasing,
(H2) \(\tau\) is non-increasing,
(H3) \(\sigma \subseteq \tau\), and
(H4) \(w[0..N]^{(N+1)@i} \leq w[0..N]^{(N+1)@j}\) for all \(i \leq j\).

**Proof** From hypothesis H4 and the definition of lexicographic ordering, we conclude that for all \(i \leq j\) and any \(m\) such that \(0 \leq m \leq N\):

LO: \(w[0..m]^{(m+1)@i} \leq w[0..m]^{(m+1)@j}\)

The proof now proceeds by induction on the number of digits in \(w\).

**Base Case:** Assume \(w\) is constructed using a single digit.

\[
w^\sigma = \begin{cases} \text{By assumption that } w \text{ is a single digit and hypothesis that } |\sigma| \geq N + 1. \end{cases}
\]

\[
w[0..0]^\sigma[0]
\leq \begin{cases} \text{By LO, since } \sigma \subseteq \tau \text{ by H3.} \end{cases}
\]

\[
w[0..0]^\tau[0]
\leq \begin{cases} \text{By assumption that } w \text{ is a single digit and hypothesis that } |\tau| \geq N + 1. \end{cases}
\]

\[
w^\tau
\]

**Induction Case:** Assume the \(w\) where \(0 \leq n < N\). We show that

\[
\{\text{From H1, } \sigma \text{ is }\}
\]

\[
\{\text{Definition of nc }\}
\]

\[
\{\text{By construction, non-decreasing.}\}
\]

\[
\{\text{By induction by } 1, 2, 3, \text{ and LO.}\}
\]

\[
\{\text{Definition of lex }\}
\]

\[
\{\text{By LO, since } \sigma[0..n]\}
\]

\[
\{\text{Transitivity with}\}
\]

\[
\{\text{Definition of nc}\}
\]

\[
\{\text{From H2, } \tau \text{ is nc }\}
\]

\[
\{\text{By construction, non-decreasing.}\}
\]

\[
\{\text{By induction by } 1, 2, 3, \text{ and LO.}\}
\]

\[
\{\text{Definition of lex }\}
\]

\[
\{\text{Transitivity with}\}
\]

\[
\{\text{Definition of } \sigma[0..n+1] = \sigma[0..n]\}
\]

\[
\{\text{w[0..n+1] is }\}
\]

\[
\square
\]
\[ \geq \sigma[i] \]. For slices \( \sigma \) and \( \tau \) such that \( \sigma[i] \leq \tau[i] \),

\[ \leq \tau[i] \).

ative order in which operations
tes that have been made to digit
lies that \( \mu_i(x) \leq \mu_i(x') \) is valid.

\[ \sigma[0..n] \] is non-decreasing. Therefore, any prefix is.

\[ \sigma[0..n] \] is non-decreasing.

\{ Definition of non-increasing. \}

\[ \alpha = (n + 1) \otimes \sigma[n + 1] \] is non-increasing.

\{ By construction of \( \sigma[0..n] \) and \( \alpha \), since (H1) \( \sigma \) is non-decreasing. \}

\[ \sigma[0..n] \subseteq \alpha. \]

\{ By induction hypothesis, since H1–H4 are satisfied due to 1, 2, 3, and LO. \}

\[ w[0..n]\tau[0..n] \leq w[0..n]^\alpha \]

\{ Definition of lexicographic order. \}

\[ w[0..n]^\sigma[0..n]w[n + 1]^\sigma[n + 1] \leq w[0..n]^\sigma w[n + 1]^\sigma[n + 1] \]

\{ By LO, since \( \sigma[n + 1] \leq \tau[n + 1] \) because (H3) \( \sigma \subseteq \tau \). \}

\[ w[0..n]^\omega w[n + 1]^\omega[n + 1] \leq w[0..n + 1]^\omega[n + 1](\omega + 2) \otimes \tau[n + 1] \]

\{ Transitivity with 5 and 6. \}

\[ w[0..n]^\tau[0..n]w[n + 1]^\tau[n + 1] \leq w[0..n + 1]^\tau[n + 1] \]

\{ Definition of non-decreasing. \}

\[ \beta = (n + 1) \otimes \tau[n + 1] \] is non-decreasing.

\{ From H2, \( \tau \) is non-increasing. Therefore, any prefix is. \}

\[ \tau[0..n] \] is non-increasing.

\{ By construction of \( \beta \) and \( \tau[0..n] \), since (H2) \( t \) is non-increasing. \}

\[ \beta \subseteq \tau[0..n] \]

\{ By induction hypothesis, since H1–H4 are satisfied due to 8, 9, 10, and LO. \}

\[ w[0..n]^\beta \leq w[0..n]^\tau[0..n] \]

\{ Definition of lexicographic order. \}

\[ w[0..n]^\beta w[n + 1]^\beta[n + 1] \leq w[0..n]^\tau[0..n]w[n + 1]^\tau[n + 1] \]

\{ Transitivity with 7 and 12. \}

\[ w[0..n]^\tau[0..n]w[n + 1]^\tau[n + 1] \leq w[0..n]^\tau[0..n]w[n + 1]^\tau[n + 1] \]

\{ \( \sigma[0..n + 1] = \sigma[0..n] \sigma[n + 1] \) and \( \tau[0..n + 1] = \tau[0..n] \tau[n + 1] \) \}

\[ w[0..n + 1]^\tau[0..n + 1] \leq w[0..n + 1]^\tau[0..n + 1] \]

\square
4 Reading to the left, writing to the right

We can now show that if the digits of \( w \) are read from right to left (i.e. \( w[n], w[n-1], \ldots, w[0] \)) but written from left to right (i.e. \( w[0], \ldots, w[n-1], w[n] \)) then only certain mixtures of values can be obtained from overlapping writes. In particular, the value read is bounded from below by the value written by the earliest write whose digit is obtained by this read.

Read-Left, Write-Right:
If (i) the sequence of values written to \( w \) is non-decreasing, (ii) digits are written from left to right, and (iii) digits are read from right to left, then the value \( w^r \) obtained by the read satisfies \( w^{(N+1) \otimes \tau[n]} \leq w^r \).

Proof We first show that \( \tau \) is non-increasing. Let \( x_i \) be the time that digit \( w[i] \) is read. Thus, \( \tau[i] = \mu_i(x_i) \) and, due to hypothesis (iii) that digits are read from right to left, \( x_n \leq x_{n-1} \leq \ldots \leq x_0 \). For any \( i, 0 \leq i < n \):

\[
\begin{align*}
\tau[i] & = \{ \text{Assumption that } \tau[i] = \mu_i(x_i) \} \\
\mu_i(x_i) & \geq \{ \text{Digits are written from left to right due to hypothesis (ii).} \} \\
\mu_{i+1}(x_{i+1}) & \geq \{ x_i \geq x_{i+1} \} \\
\mu_{i+1}(x_{i+1}) & = \{ \text{Assumption that } \tau[i] = \mu_i(x_i) \} \\
\tau[i+1] &
\end{align*}
\]

The correctness of Read-Left, Write-Right now follows from Theorem 1. Choose \( (N+1) \otimes \tau[n] \) for \( \sigma \); this choice for \( \sigma \) satisfies H1 and H3. We showed above that \( \tau \) satisfies H2. H4 is satisfied by hypothesis (i). Thus, from Theorem 1 we conclude \( w^{(N+1) \otimes \tau[n]} \leq w^r \).

5 Reading to the right, writing to the left

By reversing the order in which digits are read and written, we obtain another protocol for concurrent reading and writing. With this protocol, the value read is bounded from below by the earliest write whose digit is obtained by this read.

Read-Right, Write-Left:
If (i) the sequence of values written to \( w \) is non-decreasing, (ii) digits are written from left to right, and (iii) digits are read from right to left, then the value \( w^r \) obtained by the read satisfies \( w^{\sigma[n]} \leq w^r \).

Proof We first show that \( \sigma \) is non-decreasing. Let \( x_i \) be the time that digit \( w[i] \) is read. Thus, \( \sigma[i] = x_i \) are read from right to left, \( x_0 \)

\[
\sigma[i] = \{ \text{Assumption that } \tau \}
\mu_i(x_i)
\leq \{ \text{Digits are written from left to right due to hypothesis (ii).} \}
\mu_{i+1}(x_{i+1}) \leq \{ x_i \geq x_{i+1} \}
\mu_{i+1}(x_{i+1}) = \{ \text{Assumption that } \sigma[i] \}
\sigma[i+1]
\]

The correctness of Read-Right, Write-Left follows from Theorem 1. Choose \( (N+1) \otimes \sigma[n] \) for \( \tau \); this choice for \( \tau \) satisfies H1 and H3. We showed above that \( \sigma \) satisfies H1 and H2. H4 is satisfied by hypothesis (i). Thus, from Theorem 1 we conclude \( w^{(N+1) \otimes \sigma[n]} \leq w^r \).

There are two interesting things to note about this protocol. First, exclusive access to digits is the only synchronization required. Second, read operations and write operations do not delay each other.

6 Final remarks

This paper is now in its third revision. The informal proofs. These, like the informal proofs in the previous version of the paper contained several errors. The current version has been corrected, and its proofs are now complete.
to the right

Read-Right, Write-Left:

If (i) the sequence of values written to \( w \) is non-decreasing, (ii) digits are written from right to left, and (iii) digits are read from left to right, then the value \( w^\sigma \) obtained by any read satisfies \( w^\sigma \leq w^{(N+1) \otimes \sigma[n]} \).

Proof

We first show that \( \sigma \) is non-decreasing. Let \( x_i \) be the time that digit \( w[i] \) is read. Thus, \( \sigma[i] = \mu_i(x_i) \) and, due to hypothesis (iii) that digits are read from left to right, \( x_0 \leq x_1 \leq \ldots \leq x_n \). For any \( i, 0 \leq i < n \):

\[
\sigma[i] = \begin{cases} \text{Assumption that } \tau[i] = \mu_i(x_i). \} \\
\mu_i(x_i) \leq \begin{cases} \text{Digits are written from right to left due to hypothesis (ii).} \} \\
\mu_{i+1}(x_i) \leq \begin{cases} x_i \leq x_{i+1}. \} \\
\mu_{i+1}(x_{i+1}) \leq \begin{cases} \text{Assumption that } \sigma[i] = \mu_i(x_i). \} \\
\sigma[i+1] \end{cases} \end{cases} \end{cases} \]

The correctness of Read-Right, Write-Left now follows from Theorem 1. Choose \((N + 1) \otimes \sigma[n]\) for \( \tau \); this choice for \( \tau \) satisfies H2 and H3. We showed above that \( \sigma \) satisfies H1. H4 is satisfied by hypothesis (i). Thus, from Theorem 1 we conclude \( w^\sigma \leq w^{(N+1) \otimes \sigma[n]} \).

\[ \square \]

As before, exclusive access to digits is the only synchronization required, and operations are never delayed.

6 Final remarks

This paper is now in its third revision. The first version contained simple and informal proofs. These, like the proof of Theorem 1 given above, used induction on the number of digits in a word. Unfortunately, the proofs were wrong — the informality let details slip through the cracks. The second version of the paper contained correct and formal versions of those proofs. A total of four lemmas were required — two lemmas for each protocol— although the two pairs of lemmas had proofs that were disturbingly similar. Theorem 1 of the current version of the paper generalizes two of those lemmas, and its proof results from combining the proofs of those two lemmas.
Acknowledgments: David Gries read and commented on many earlier versions of this paper. Jay Misra pointed out the errors in the first version of the paper and proposed the statement of Theorem 1 along with a (long) proof. Avoiding a case analysis in that proof led to the proof finally given above.

REFERENCES


Fred B. Schneider,
Department of Computer Science,
Cornell University,
4130 Uplson Hall,
Ithaca, New York 14853–7501,
U.S.A.

Goodbye Juncti

Carel S. Scholten

In [2] the notions conjunctivity and disjunctivity are introduced. For the conjunctivity. Prima
(0) \( f \) is conjunctive over a set \( V \)
\( \{ f | (A X : X \in V : X) \} \)
In the above, the infix dot denotes universal quantification. Conjunctivity properties of \( f \) are:
(1) \( f \) is universally conjunctive
(2) \( f \) is positively conjunctive
(3) \( f \) is conjunctive over a certain type.

In [1] Dijkstra raises the question of whether a good way to introduce, more formally, the two implicit
hand side of (0) separately.

Well, to begin with, the notion of "conjunction" and "disjunction" is faithful to the originalopes:
Thank you" in the title of this note.

Nevertheless, let us follow Dijkstra in working these two properties of \( f \) and \( V \):
(1) \( f \) is universally conjunctive
(2) \( f \) is positively conjunctive

For (1), no special name is proposed for the property satisfied by all pairs \( f \), \( V \) satisfies
any \( f \), the three assertions "(1) \( f \) is conjunctive over \( V \)" and "(1) \( f \) is monotonic" are equi-
