Reading the Linearizability paper of Herlihy and Wing

Uri Abraham *

May 28, 2013

Abstract

We read Herlihy and Wing important paper on the concept of linearizability, and reconstruct that text in the Tarskian event structure paradigm. With linearizability defined in terms of Tarskian system executions we obtain a different correctness proof for the queue algorithm that Herlihy and Wing present in that paper.

1 Definition of linearizability by means of Tarskian system executions

The definition that we give here for linearizability is virtually the same as the definition that was originally given by Herlihy and Wing in their very influential paper¹, but since we use Tarskian system executions rather than histories in our definition, the two definitions may superficially appear to be different. We will argue in this article that (with some evident exaggeration) the authors of the “Linearizability” paper have spoken in Tarskian prose even while presenting their linearizability theory in terms of states and histories.

Recall the definition of moment based system execution languages. Such a language contains the sort Moment, and there are two functions Left_End and Right_End so that, for any event e, \([\text{Left}_\text{End}(e), \text{Right}_\text{End}(e)]\) is the temporal extension of e (an interval in the linearly ordered set Moment). Assume a class \(\mathcal{N}\) of system executions that interpret L so that in every


$N \in \mathcal{N}$ the events are linearly ordered by the precedence relation $<^N$ of that structure. An arbitrary system execution $M$ that interprets $L$ (for which $<^M$ is not necessarily a linear ordering) is linearizable (with respect to $\mathcal{N}$) if there is a linear ordering $<^*$ of the set of events of $M$ so that $<^*$ extends $<^M$ and so that if we replace $<^M$ with $<^*$ in $M$ then the resulting structure $N$ is in $\mathcal{N}$.

The following example should clarify this notion of linearizability. Recall the language $L_{\text{queue}}$ which we used to specify serial queues. There were two sorts $\text{Event}$ and $\text{Data}$, two predicates $\text{enq}$ and $\text{deq}$, a function $\text{Val}$, and most importantly the function $\gamma$ that yielded for every successful dequeue event $d$ its corresponding enqueue event $\gamma(d)$. (There were additional items in $L_{\text{queue}}$ but these were the most important). Recall also that we defined a sentence in that language (let’s call it $\psi$) which specifies the behavior of serial queues. The main ingredients of this statement were that the enqueue and dequeue events are linearly ordered by $<$, that $\text{Val}(d) = \text{Val}(\gamma(d))$ for any dequeue event $d$ (unless $d$ returns the “empty” value), that $\gamma(d) < d$, and that $\gamma$ is order preserving and one-to-one (which is a reflection of the FIFO property). Let $\mathcal{N}$ be the set of all system executions $N$ that interpret $L_{\text{queue}}$ and which satisfy $\psi$. That is, $\mathcal{N}$ is the set of all system executions in which the queue operations correctly represent a serial FIFO regime. Then an arbitrary system execution interpretation of $L_{\text{queue}}$ is linearizable if its partial ordering of events can be augmented to yield a structure in $\mathcal{N}$.

## 2 The Herlihy and Wing queue algorithm

The Herlihy and Wing queue algorithm is in figure 1. The algorithm implements a distributed queue that allows Enqueue and Dequeue operations by concurrently operating processes.

The algorithm assumes an infinite array $\text{items}$ indexed with natural numbers: $\text{items}[1], \text{items}[2], \ldots$, and initially all entries $\text{items}[i]$ have the “empty” value $\ast$ (which is not a $\text{Data}$ value). Every cell $\text{items}[i]$ is a serial register. We also assume a global variable $\text{back}$ of type natural number which is initially 1. The role of $\text{back}$ is to hold the index of the first cell in items that is available for the next $\text{Enqueue}$ operation. Operation $\text{Enqueue}(x)$ (where $x$ is the $\text{Data}$ value with which the operation is invoked) begins with an atomic action $i := \text{INC}(\text{back})$ which assigns to $i$ the value of $\text{back}$ and then increases $\text{back}$ by one. The algorithm would not work properly if this action is split into separate assign and increase parts, but we do assume an atomic operation which ensures that operation $\text{INC}$ is executed instantly.
Enqueue($x$)
1. $i := \text{INC}(\text{back});$
2. $\text{items}[i] := x.$

Dequeue
while true
1. $\text{range} := \text{back} - 1;$
2. for $i := 1$ to $\text{range}$ do
   $x := \text{swap}(\text{items}[i], *);$ 
   if $x \neq *$ then return $x$

Figure 1: The Hrlihy and Wing queue protocol.

The Dequeue operation consists of one or more "cycles". A cycle is an execution of lines 1 and 2. A successful cycle is a cycle that returns. So a Dequeue operation may consist of a finite number of cycles, the last being successful, or of an infinite number of non-successful cycles. In executing line 1 of a cycle, variable back is read and the value range $- 1$ is assigned to local variable range of the executing process. The swap operation in line 2 is an atomic operation that reads the value of items[$i$], assigns the value obtained into local variable $x$, and replaces this value with *. This is an atomic action.

We shall examine now an arbitrary execution of this algorithm, and state some properties that are rather easy to derive informally. These properties are collected in Figure 2, and we will prove that they imply that the queue is linearizable. The properties refer to high-level events, and so we first define these events. An Enqueue operation is an execution of lines 1 and 2 of the Enqueue code by one of the concurrent processes. If $E$ is an Enqueue operation then inc($E$) is the execution of line 1 in $E$. Val(inc($E$)) is the value of $i$ that is obtained in this action. Then asn($E$) is the execution of line 2, in which $x = \text{Val}(E)$ is assigned to items[Val(inc($E$))].

There are two possibilities for a Dequeue operation execution $D$. $D$ is not successful if it is non-terminating, and in this case it contains an infinite number of cycles. When $D$ is terminating only its last cycle $C$ is important for the proof. We denote with $b(D)$ the read action in $C$ of the back variable, and we let $r = \text{Val}(b(D))$ be the value obtained in this read. Then $s(D)$ denotes the last swap action in $C$, which is the action in which the “empty” token * is written on items[$r$] instead of the Data value $x = \text{Val}(D)$ which is returned.

Since a Dequeue operation, even when successful, may contain unsuccessful cycles that are not relevant for the proof we define $\text{Deq}$ events which
are just the successful cycles of Dequeue events. So if $D$ is any Dequeue event then its last cycle is a Deq event. When $D$ is a Deq event then $b(D)$ and $s(D)$ denote the read of back and the swap actions in $D$, so that $D = \{b(D), s(D)\}$. And $\text{begin}(D)$, $\text{end}(D)$ are the Moments at which $b(D)$ and $s(D)$ are executed.

We also define Enq events, which are just the Enqueue operation executions. So any Enq event $E$ has the form $E = \{\text{inc}(E), \text{asn}(E)\}$. (And $\text{begin}(E)$, $\text{end}(E)$ are their Moment values.)

We define the function $\gamma$ on the terminating Deq events. If $D$ is a terminating Deq event and $s(D)$ its (last and successful) swap action executed on items $[i]$, then the Data value $x$ obtained must have been assigned to items $[i]$ by some Enq operation $E$ in executing the write operation $\text{asn}(E)$. Note that $E$ is uniquely determined by index $i$ since different Enq events access different indexes. We then define $E = \gamma(D)$.

The HW properties enumerated in Figure 2 are expressed in the $L_{\text{queue}}$ language. We use the letter $D$ to denote Deq events (which are successful), and $E$ refers to Enq events. Since the domain of $\gamma$ is the set of Deq events, when we write $\gamma(X)$ it is implicitly evident that $\text{Deq}(X)$ holds.

In class we gave intuitive arguments to explain why the HW properties hold in every execution of the algorithm. Our main aim is to show that they imply linearizability. This is done in the following section ??.

Only after seeing the relevance of these properties we shall return and prove more formally that they hold.

### 2.1 High level properties and their linearizability consequence

Our aim is to prove that in any Tarskian system execution in which the HW properties 1–3 hold, a linearizable queue is implemented.

**Theorem 2.1** In the context of Moment Based System Executions, properties HW imply that there exists a linear ordering $\prec^*$ that extends $\prec$ and which satisfies the following.

1. $\gamma(D) \prec^* D$ for every Deq event $D$.

2. The function $\gamma$ is $\prec^*$-order preserving. That is, for every successful Deq events $D_1, D_2$: $D_1 \prec^* D_2$ iff $\gamma(D_1) \prec^* \gamma(D_2)$.

A main idea for the proof is to linearly order the Deq events by the ordering of their end-points. That is, we intend to define $D_1 \prec^* D_2$ iff $\text{end}(D_1) < \text{end}(D_2)$. This decision immediately entails the ordering of the
1. Every event is either a \( Deq \) or else an \( Enq \) event. All \( Enq \) events are terminating. If \( D \) is a terminating \( Deq \) event then \( \gamma(D) \) is defined, it is an \( Enq \) event and \( Val(\gamma(D)) = Val(D) \).

2. The functions \( begin \), \( end \) are defined on the events and yield Moment values. These are one-to-one functions. The \( \gamma \) function is also one-to-one.

3. If \( E = \gamma(D) \) then \( Val(E) = Val(D) \) and

\[
begin(E) < begin(D), \text{ and } end(E) < end(D). \tag{1}
\]

4. If \( E_0 < \gamma(D) \) (where \( E_0 \) is an \( Enq \) event) then, for some \( Deq \) event \( D_0 \), \( E_0 = \gamma(D_0) \) and \( end(D_0) < end(D) \).

A stronger form holds: If \( E = \gamma(D) \) and \( E_0 \) is an \( Enq \) event so that \( E_0 < D \) and \( begin(E_0) < begin(E) \), then for some \( Deq \) event \( D_0 \)

\[
E_0 = \gamma(D_0) \text{ and } end(D_0) < end(D).
\]

Figure 2: The HW list of properties of the high level events of the Herlihy and Wing queue algorithm.
corresponding Enq events. That is, we shall request that for Enq events $E_1$ and $E_2$ of the form $E_\ell = \gamma(D_\ell)$ (for $\ell = 1, 2$) the equivalence $E_1 \prec^* E_2$ iff $\text{end}(D_1) < \text{end}(D_2)$ holds. In addition, we shall demand that $\gamma(D) \prec^* D$ for every successful Deq event $D$. And, of course, we demand that the relation $\prec^*$ extends the given partial ordering $\prec$. Combining these demands we get a relation $\prec$ on the Enq/Deq events which is not necessarily an ordering relation. The main mission in the proof of Theorem 2.1 is the proof that $\prec$ contains no cycles and hence that the transitive closure of $\prec$ is indeed a partial ordering. This is achieved in a series of lemmas.

We are going to define five relations on the events, $\prec_i$ for $i = 1, \ldots, 5$, and then define $\prec$ as their union. That is $A \prec B$ iff $A \prec_i B$ for some $i = 1, \ldots, 5$. The main step in the proof is showing that $\prec$ has no cycles and hence that its transitive closure $\prec^*$ is an irreflexive relation (and thus a partial ordering). Now $\prec^*$ is not necessarily a linear ordering, but a simple extension of $\prec^*$ yields the desired linearization.

Define the following relations:

1. For any events $A$ and $B$: $A \prec_1 B$ iff $A < B$.
2. For Enq and Deq events $E$ and $D$ define $E \prec_2 D$ iff $E = \gamma(D)$.
3. For Deq events $D_1$ and $D_2$ define $D_1 \prec_3 D_2$ iff $\text{end}(D_1) < \text{end}(D_2)$.
4. For Enq events $E_1$ and $E_2$ define $E_1 \prec_4 E_2$ iff for some Deq events $D_1$ and $D_2$ we have that $E_1 = \gamma(D_1)$, $E_2 = \gamma(D_2)$, and $\text{end}(D_1) < \text{end}(D_2)$.
5. For Enq events $E_1$ and $E_2$ define $E_1 \prec_5 E_2$ iff $E_1$ is in the range of $\gamma$, but $E_2$ is not.

Note that these relations are not disjoint and $\prec$ may have a non-empty intersection with any of the other relations. For example, if $D_1, D_2$ are Deq events and $D_1 \prec D_2$ then also $D_1 \prec_3 D_2$. Relations $\prec_i$ for $i > 1$ however, are mutually exclusive.

Define $X \prec Y$ iff $X \prec_i Y$ for some $i = 1, \ldots, 4$. Let $\prec^*$ be the transitive closure of $\prec$. That is, $X \prec^* Y$ iff there exists a sequence $X_0, \ldots, X_n$ of events (with $n > 0$) where $X_0 = X$ and $X_n = Y$ and so that $X_k \prec X_{k+1}$ for every $k < n$. The following sequence of lemmas will finally prove that $\prec^*$ is an irreflexive relation.

**Lemma 2.2** If $D_1, D_2$ are Deq events then $D_1 \prec D_2$ iff $D_1 \prec_3 D_2$
Proof. \( D_1 \prec_1 D_2 \) implies \( D_1 \prec_3 D_2 \), and relations \( \prec_2 \) and \( \prec_4 \) are not applicable. \( \text{q.e.d.} \)

Note that \( \prec_2 \) is a transitive relation simply because it is never the case that \( X \prec_2 Y \prec_2 Z \) (which would entail that \( Y \) is both an ENQ and a DEQ event. Relation \( \prec_3 \) is also transitive (because \( \gamma \) is one-to-one). Similarly, all other \( \prec_i \) relations are transitive, and the following is easy to prove.

Lemma 2.3 Each \( \prec_i \) for \( i = 1, \ldots, 5 \) is an irreflexive and transitive relation. Relation \( \prec_3 \) is in fact a linear ordering of the terminating \( \text{Deq} \) events which extends the \( < \) relation over these events.

Proof. Suppose \( E_0 < E_1 \) are \( \text{Enq} \) events and \( E_1 \) is in the range of \( \gamma \) then \( E_0 \prec_4 E_1 \) (and in particular \( E_0 \) is also in the range of \( \gamma \)).

Proof. Suppose \( E_0 \) and \( E_1 \) are both \( \text{Enq} \) events, \( E_0 \prec E_1 \) and \( E_1 = \gamma(D_1) \). Relations \( \prec_2 \) and \( \prec_3 \) clearly do not apply to \( E_0 \prec E_1 \) since these relations never relate two \( \text{Enq} \) events. But \( E_0 \prec_3 E_1 \) is also impossible since \( E_1 \) is in the range of \( \gamma \). This leaves two possibilities: \( E_0 < E_1 \) and \( E_0 \prec_4 E_1 \).

To prove the lemma we show that \( E_0 < E_1 \) implies \( E_0 \prec_4 E_1 \). Since \( E_1 = \gamma(D_1) \), property 3 of the HW list implies (in case \( E_0 < E_1 \)') that there exists some \( \text{Deq} \) event \( D_0 \) with \( E_0 = \gamma(D_0) \) and \( \text{end}(D_0) < \text{end}(D_1) \). That is \( E_0 \prec_4 E_1 \) as required. \( \text{q.e.d.} \)

Lemma 2.5 The \( \prec \) relation has no cycles of length two.

Proof. Since \( \prec \) is irreflexive, it is never the case that \( X \prec X \). So we have to show that there are no two events \( X \neq Y \) such that \( X \prec_i Y \prec_j X \) for some \( i \) and \( j \). To prove this let’s check the different possibilities. First, \( i = j \) is impossible since \( \prec_i \) is transitive, and \( X \prec_i Y \prec_i X \) would contradict the irreflexivity of \( \prec_i \). So it must be that \( X \prec_i Y \prec_j X \) where \( i \neq j \).

1. Suppose both \( X \) and \( Y \) are both \( \text{Deq} \) events. We noted already (Lemma 2.2) that if \( D_1, D_2 \) are \( \text{Deq} \) events then \( D_1 \prec D_2 \) implies that \( D_1 \prec_3 D_2 \). Hence \( X \prec Y \prec \) would bring us back to \( X \prec_3 Y \prec_3 X \) which we ruled out (by transitivity and irreflexivity of \( \prec_3 \)).

2. Suppose that \( X \) and \( Y \) are both \( \text{Enq} \) events. Say \( E_1 \prec_i E_2 \prec_j E_1 \). Relations \( \prec_2 \) and \( \prec_3 \) do not apply on two \( \text{Enq} \) events, and so we have to check all other possibilities for \( i, j \in \{1, 4, 5\} \) when \( i \neq j \).

(a) Suppose that \( \{i, j\} = \{1, 4\} \). Say \( E_1 \prec_4 E_2 < E_1 \). So there are \( \text{Deq} \) events \( D_1 \) and \( D_2 \) with \( E_1 = \gamma(D_1) \), \( E_2 = \gamma(D_2) \) and
end(D_1) < end(D_2). By property 3 of HW applied to E_2 < γ(D_1) there is a Deq event D_0 such that E_2 = γ(D_0) and end(D_0) < end(D_1). That is, E_2 ≺_4 E_1, and we are back to case i = j.

(b) Now suppose that \{i, j\} = \{1, 5\}. Say E_1 ≺_5 E_2 < E_1. So E_1 is in the domain of γ but E_2 is not. But this case is impossible by Lemma 2.4.

(c) Finally suppose that \{i, j\} = \{4, 5\}. Say E_1 ≺_4 E_2 ≺_5 E_1. But this is also impossible since E_1 ≺_4 E_2 implies that both of E_1 and E_2 are in the range of γ.

3. Suppose now that one of X and Y is an Enq event and the other a Deq event: E < D < E. An inspection of relations ≺_4 shows that D < E implies D < E. And E < D implies E < D or E ≺_2 D < E. Surely it is not the case that E < D < E, and so it remains to consider E ≺_2 D < E. But E = γ(D) would imply end(E) < end(D) by property 2 of the HW list, in contradiction to D < E.

q.e.d

**Lemma 2.6** Relation ≺ has no cycles.

Proof. Assume on the contrary that there is a cycle, and take a cycle of minimal length n. In the previous lemma we ruled out the case that n = 2, and so we assume that a cycle of minimal length n > 2 exists: X_1 ≺ X_2 ··· ≺ X_n ≺ X_1.

1. In case all X_i are Deq event, we get (with Lemma 2.2) that end(X_i) < end(X_{i+1}) and end(X_n) < end(X_1) which is impossible since there are no cycles in the ≺ ordering.

2. In case all X_i are Enq events there are three possibilities. If no X_i is in the range of γ, then X_i ≺ X_{i+1} implies X_i < X_{i+1} and we reach a contradiction since ≺ is an ordering relation. If all X_i’s are in the range of γ then we have a cycle in the ≺_4 relation (Lemma 2.4), which would contradict the transitivity and irreflexivity of ≺_4. Finally, if some of the X_i’s are in the range of γ and some are not, then we can find two successive Enq on the cycle, E ≺ E' where E is not in the range and E' is in the range of γ. But then we also have E' ≺_5 E by definition of ≺_5 in contradiction to our earlier result that ≺ has no cycles of length two.
3. So we are left with the case that there are both Enq and Deq events in our minimal cycle. We can find two consecutive members on the cycle a Deq event followed by an Enq event. Say $X_2 = D$ and $X_3 = E$ form such a pair. Then $D \prec E$ implies $D < E$ since no other relation applies. Consider the possibilities for $X_1$:

(a) If $X_1 = E_1$ is an Enq event, then $E_1 \prec D$ allows that either $E_1 < D$ or $E_1 \prec_2 D$. In the first case, $E_1 < D < E$ yields $E_1 < E$ and the cycle can be shortened in contradiction to its minimality. But in the second case, $E_1 = \gamma(D)$ implies $\text{end}(E_1) < \text{end}(D)$ which in light of $D < E$ yields $E_1 < E$ which again shortens the cycle.

(b) $X_1 = D_1$ is a Deq event. So the situation is that $D_1 \prec D < E$. But $D_1 \prec D$ implies that $D_1 < D$ or $D_1 \prec_3 D$. The first possibility leads again to a shorter cycle, and so we have that $D_1 \prec_3 D$. But this means that $\text{end}(D_1) < \text{end}(D)$, and hence $D_1 < E$ follows which again can be used to shorten the cycle.

q.e.d

So relation $\prec$ has no cycles, and its transitive closure $\prec^*$ is thence an irreflexive ordering relation. Any ordering relation has an extension to a linear ordering and if we extend $\prec^*$ to a linear relation $<^*$ we conclude the proof of theorem 2.1.