Safety of Deferred Update in Transactional Memory

Hagit Attiya\textsuperscript{1} Sandeep Hans\textsuperscript{1} Petr Kuznetsov\textsuperscript{2,3} Srivatsan Ravi\textsuperscript{2}

\textsuperscript{1}Department of Computer Science, Technion
\textsuperscript{2}Telekom Innovation Laboratories, TU Berlin
\textsuperscript{3}Télécom ParisTech

Abstract

Transactional memory allows the user to declare sequences of instructions as speculative transactions that can either commit or abort. If a transaction commits, it appears to be executed sequentially, so that the committed transactions constitute a correct sequential execution. If a transaction aborts, none of its instructions can affect other transactions.

The popular criterion of opacity requires that the views of aborted transactions must also be consistent with the global sequential order constituted by committed ones. This is believed to be important, since inconsistencies observed by an aborted transaction may cause a fatal irrecoverable error or waste of the system in an infinite loop. Intuitively, an opaque implementation must ensure that no intermediate view a transaction obtains before it commits or aborts can be affected by a transaction that has not started committing yet, so called deferred-update semantics.

In this paper, we intend to grasp this intuition formally. We propose a variant of opacity that explicitly requires the sequential order to respect the deferred-update semantics. Unlike opacity, our property also ensures that a serialization of a history implies serializations of its prefixes. Finally, we show that our property is equivalent to opacity if we assume that no two transactions commit identical values on the same variable, and present a counter-example for scenarios when the “unique-write” assumption does not hold.

1 Introduction

Resolving conflicts in an efficient and consistent manner is the most challenging task in concurrent software design. Transactional memory (TM) \cite{11,18} addresses this challenge by offering an interface in which sequences of shared-memory instructions can be declared as speculative transactions. The underlying idea, borrowed from databases, is to treat each transaction as an atomic event: a transaction may either commit in which case it appears as executed sequentially, or abort in which case none of its update instructions affect other transactions. The user can therefore design software having only sequential semantics in mind and let the memory take care of conflicts resulting from potentially concurrent executions.

In databases, a correct implementation of concurrency control should guarantee that committed transactions constitute a serial (or sequential) execution \cite{9}. On the other hand, uncommitted transactions can be aborted without invalidating the correctness of committed ones. (In the literature on databases, the latter feature is called recoverability.)

In the TM context, intermediate states witnessed by an incomplete transaction may affect the application through the outcome of its read operations. If the intermediate state is not consistent with any sequential execution, the application may experience a fatal irrecoverable
error or sink in an infinite loop. The correctness criterion of opacity \cite{7,8} addresses this issue by requiring the states observed by uncommitted transactions to be consistent with a global serial execution constituted by committed ones (a serialization).

An opaque TM implementation must, intuitively, ensure that no transaction can read from a transaction that has not started committing yet. This is usually referred to as the deferred-update semantics, and it was in fact explicitly required in some representations of opacity \cite{6}. The motivation of this paper is to capture this intuition formally.

We present a new correctness criterion called du-opacity. Informally, a du-opaque (possibly, non-serial) execution must be indistinguishable from a totally-ordered execution, with respect to which no transaction reads from a transaction that has not started committing.

We further check if our correctness criterion is a safety property, as defined by Owicki and Lamport \cite{17}, Alpern and Schneider \cite{2} and refined by Lynch \cite{16}. We show that du-opacity is prefix-closed: every prefix of a du-opaque history is also du-opaque. We also show that du-opacity is, under certain restrictions, limit-closed. More precisely, assuming that, in an infinite execution, every transaction completes each of the operations it invoked (but possibly neither commits nor aborts), the infinite limit of any sequence of ever extending du-opaque histories is also du-opaque. To prove that such an implementation is du-opaque, it is thus sufficient to prove that all its finite histories are du-opaque. To the best of our knowledge, this paper contains the first non-trivial proof of limit-closure for a TM correctness property. We further show that any du-opaque serialization of a history implies a serialization of any of its prefixes that maintains the original read-from relations, which is instrumental in the comparison of du-opacity with opacity.

Opacity, as defined in \cite{8}, reduces correctness of an infinite history to correctness of all its prefixes, and thus is limit-closed by definition. In fact, we show that extending opacity to infinite histories in a non-trivial way (i.e., requiring that even infinite histories should have proper serializations), does not result in a limit-closed property. We observe that opacity does not preclude scenarios in which a transaction reads from a future transaction (cf. example in Figure 4), and, thus, our criterion is strictly stronger than opacity. Surprisingly, this is true even if we assume that all transactional operations are atomic, which somewhat attenuates earlier attempts to forcefully introduce the deferred-update in the definition of opacity for atomic operations \cite{6}. However, we show that opacity and du-opacity are equivalent if we assume that no two transactions try to commit identical values on the same data item.

We believe that these results improve our understanding of the very notion of correctness in transactional memory. Our correctness criterion explicitly declares that a transaction is not allowed to read from a transaction that has not started committing yet, and we conjecture that it is simpler to verify. We present the first non-trivial proof for both limit- and prefix-closure of TM histories, which is quite interesting in its own right, for it enables reasoning about possible serializations of an infinite TM history based on serializations of its prefixes.

The paper is organized as follows. In Section 2, we introduce our basic model definitions and recall the notion of safety \cite{2,16,17}. In Section 3, we define our criterion of du-opacity and show that it is prefix-closed and under certain restrictions, a limit-closed property. In Section 4, we prove that du-opacity is a proper subset of the original notion of opacity \cite{8}, and that it coincides with du-opacity under the “unique-writes” condition.

2 Model

A transactional memory (in short, TM) supports atomic transactions for reading and writing on a set of transactional objects (in short, t-objects). A transaction is a sequence of accesses (reads or writes) to t-objects; each transaction $T_k$ has a unique identifier $k$. 
A transaction $T_k$ may contain the following t-operations, each being a matching pair of invocation and response events:

1. $\text{read}_k(X)$ returns a value in some domain $V$ or a special value $A_k \notin V$ (abort);
2. $\text{write}_k(X,v)$, for a value $v \in V$, returns $ok_k$ or $A_k$;
3. $\text{tryC}_k$ returns $C_k \notin V$ (commit) or $A_k$; and
4. $\text{tryA}_k$ returns $A_k$.

The read set (resp., the write set) of a transaction $T_k$, denoted $Rset(T_k)$, is the set of t-objects that $T_k$ reads in $H$; the write set of $T_k$, denoted $Wset(T_k)$, is the set of t-objects $T_k$ writes to in $H$.

We consider an asynchronous shared-memory system in which processes communicate via transactions. A TM implementation provides processes with algorithms for implementing $\text{read}_k$, $\text{write}_k$, $\text{tryC}_k()$ and $\text{tryA}_k()$ of a transaction $T_k$.

A history of a TM implementation is a (possibly infinite) sequence of invocation and response events of t-operations.

For every transaction identifier $k$, $H|k$ denotes the subsequence of $H$ restricted to events of transaction $T_k$. If $H|k$ is non-empty, we say that $T_k$ participates in $H$, and let $txns(H)$ denote the set of transactions that participate in $H$. In an infinite history $H$, we assume that each $T_k \in txns(H)$, $H|k$ is finite; i.e., transactions do not issue an infinite number of t-operations.

Two histories $H$ and $H'$ are equivalent if $txns(H) = txns(H')$ and for every transaction $T_k \in txns(H)$, $H|k = H'|k$.

A history $H$ is sequential if every invocation of a t-operation is either the last event in $H$ or is immediately followed by a matching response.

A history is well-formed if for all $T_k$, $H|k$ is sequential and has no events after $A_k$ or $C_k$.

We assume that all histories are well-formed, i.e., the client of the transactional memory never invokes a t-operation before receiving a response from the previous one and does not invoke any t-operation $\text{op}_k$ after receiving $C_k$ or $A_k$. We also assume, for simplicity, that the client invokes a $\text{read}_k(X)$ at most once within a transaction $T_k$. This assumption incurs no loss of generality, since a repeated read can be assigned to return a previously returned value without affecting the history’s correctness.

A transaction $T_k \in txns(H)$ is complete in $H$ if $H|k$ ends with a response event. The history $H$ is complete if all transactions in $txns(H)$ are complete in $H$.

A transaction $T_k \in txns(H)$ is t-complete if $H|k$ ends with $A_k$ or $C_k$; otherwise, $T_k$ is t-incomplete. $T_k$ is committed (resp., aborted) in $H$ if the last event of $T_k$ is $C_k$ (resp., $A_k$). The history $H$ is t-complete if all transactions in $txns(H)$ are t-complete.

For t-operations $\text{op}_k, \text{op}_j$, we say that $\text{op}_k$ precedes $\text{op}_j$ in the real-time order of $H$, denoted $\text{op}_k \prec_{RT} \text{op}_j$, if the response of $\text{op}_k$ precedes the invocation of $\text{op}_j$.

Similarly, for transactions $T_k, T_m \in txns(H)$, we say that $T_k$ precedes $T_m$ in the real-time order of $H$, denoted $T_k \prec_{RT} T_m$, if $T_k$ is t-complete in $H$ and the last event of $T_k$ precedes the first event of $T_m$ in $H$. If neither $T_k \prec_{RT} T_m$ nor $T_m \prec_{RT} T_k$, then $T_k$ and $T_m$ overlap in $H$. A history $H$ is t-sequential if there are no overlapping transactions in $H$.

For simplicity of presentation, we assume that each history $H$ begins with an “imaginary” transaction $T_0$ that writes initial values to all t-objects and commits before any other transaction begins in $H$.

Let $H$ be a t-sequential history. For every operation $\text{read}_k(X)$ in $H$, we define the latest written value of $X$ as follows:
1. If \( T_k \) contains a \( \text{write}_k(X,v) \) preceding \( \text{read}_k(X) \), then the latest written value of \( X \) is the value of the latest such write to \( X \).

2. Otherwise, if \( H \) contains a \( \text{write}_m(X,v) \), \( T_m \) precedes \( T_k \), and \( T_m \) commits in \( H \), then the latest written value of \( X \) is the value of the latest such write to \( X \) in \( H \). (This write is well-defined since \( H \) starts with \( T_0 \) writing to all \( t \)-objects.)

We say that \( \text{read}_k(X) \) is \textit{legal} in a t-sequential history \( H \) if it returns the latest written value of \( X \), and \( H \) is \textit{legal} if every \( \text{read}_k(X) \) in \( H \) that does not return \( A_k \) is legal in \( H \).

**Definition 1** ([2][16]). A property \( P \) is a set of (transactional) histories. A property \( P \) is a \textit{safety} property if it satisfies:

1. Prefix-closure: every prefix \( H' \) of a history \( H \in P \) is also in \( P \) and

2. Limit-closure: for any infinite sequence of finite histories \( H^0, H^1, \ldots \) such that for all \( i \), \( H^i \in P \) and \( H^i \) is a prefix of \( H^{i+1} \), the infinite history that is the limit of the sequence is also in \( P \).

Notice that the set of histories produced by a TM implementation \( M \) is prefix-closed. Therefore, every infinite history of \( M \) is the limit of an infinite sequence of ever-extending finite histories of \( M \). Thus, to prove that \( M \) satisfies a safety property \( P \), it is enough to show that all finite histories of \( M \) are in \( P \). Indeed, limit-closure of \( P \) then implies that every infinite history of \( M \) is also in \( P \).

### 3 DU-Opacity

In this section, we introduce our correctness criterion, \textit{du-opacity}, and prove that a restriction of it is a limit-closed property.

**Definition 2.** Let \( H \) be any history. A completion of \( H \), denoted \( \hat{H} \), is a history derived from \( H \) as follows:

- for every incomplete t-operation \( op_k \) of transaction \( T_k \in \text{txns}(H) \) in \( H \), if \( op_k = \text{read}_k \lor \text{write}_k \lor \text{try}_A_k() \), insert \( A_k \) somewhere after the invocation of \( op_k \); otherwise, if \( op_k = \text{try}_C_k() \), insert \( C_k \) or \( A_k \) somewhere after the last event of \( T_k \).

- for every complete transaction \( T_k \in \text{txns}(H) \) that is not t-complete, insert \( \text{try}_C_k \cdot A_k \) after the last event of \( T_k \).

Let \( H \) be any history and \( S \) be a legal t-complete t-sequential history that is equivalent to some completion of \( H \). Let \( <_S \) be the total order on transactions in \( S \).

For any \( \text{read}_k(X) \) that does not return \( A_k \), let \( S^{k,X} \) denote the prefix of \( S \) up to the response of \( \text{read}_k(X) \) and \( H^{k,X} \) denotes the prefix of \( H \) up to the response of \( \text{read}_k(X) \). Let \( S^k_X \) denote the subsequence of \( S^{k,X} \) derived by removing from \( S^{k,X} \) the events of all transactions \( T_m \in \text{txns}(H) \) such that \( H^{k,X} \) does not contain an invocation of \( \text{try}_C_m() \). We refer to \( S^k_X \) as the local serialization for \( \text{read}_k(X) \) with respect to \( H \) and \( S \).

We are now ready to present our correctness condition, \textit{du-opacity}.

**Definition 3.** A history \( H \) is \textit{du-opaque} if there is a legal t-complete t-sequential history \( S \) such that

(1) there exists a completion of \( H \) that is equivalent to \( S \), and
We then say that extending sequence is not du-opaque.

Figure 1: A du-opaque history \( H \); there exists a serialization \( S \) of \( H \) such that each t-read in \( S \) has a legal local serialization with respect to \( H \) and \( S \).

Figure 2: Each finite prefix of the history is du-opaque, but the infinite limit of the ever-extending sequence is not du-opaque.

\[(2) \text{ for every pair of transactions } T_k, T_m \in \text{txns}(H), \text{ if } T_k \sim^\text{RT}_H T_m, \text{ then } T_k \prec_S T_m, \text{ i.e., } S \text{ respects the real-time ordering of transactions in } H, \text{ and }\]

\[(3) \text{ each read}_k(X) \text{ in } S \text{ that does not return } A_k \text{ is legal in } S^k_{H,X}.\]

We then say that \( S \) is a (du-opaque) serialization of \( H \). Let seq\((S)\) denote the sequence of transactions in \( S \) and seq\((S)\)[\(k\)] denote the \( k\)th transaction in this sequence.

Informally, a history \( H \) is du-opaque if there exists a legal t-sequential history \( S \) that is equivalent to \( H \), respects the real-time ordering of transactions in \( H \) and every t-read is legal in its local serialization with respect to \( H \) and \( S \). The third condition reflects the implementation’s deferred-update semantics, i.e., the legality of a t-read in a serialization does not depend on transactions that start committing after the response of the t-read.

An example of a du-opaque history \( H \) is presented in Figure 1. Let \( S \) be the t-complete t-sequential history such that seq\((S)\) = \( T_2, T_3, T_1, T_4 \) and \( S \) is equivalent to \( H \) (\( H \) is its own completion). It is easy to see that \( S \) is legal and respects the real-time order of transactions in \( H \). We now need to prove that each t-read performed in \( S \) has a local serialization with respect to \( H \) in \( S \) that is legal. Consider read\(_1\)(\( X \)) in \( S \); since \( T_2 \) is t-complete in \( H^{1,X} \), it follows that read\(_1\)(\( X \)) is legal in \( T_2 \cdot \text{read}_1(X) \) (local serialization for read\(_1\)(\( X \)) with respect to \( H \) and \( S \)). Similarly, since \( T_1, T_2, T_3 \) are t-complete in \( H^{4,X} \), read\(_4\)(\( X \)) is legal in \( T_2 \cdot T_3 \cdot T_1 \cdot \text{read}_4(X) \) (local serialization for read\(_4\)(\( X \)) with respect to \( H \) and \( S \)). Thus, \( S \) is a du-opaque serialization of \( H \).

For a history \( H \), let \( H^i \) be the finite prefix of \( H \) of length \( i \), i.e., consisting of the first \( i \) events of \( H \). Now we show a property of du-opaque histories that is going to be instrumental in the rest of the paper.

**Lemma 1.** Let \( H \) be a du-opaque history and \( S \) be a serialization of \( H \). For any \( i \in \mathbb{N} \), there exists a serialization \( S^i \) of \( H^i \), such that seq\((S^i)\) is a subsequence of seq\((S)\).

**Proof.** Given \( H, S \) and \( H^i \), we construct a t-complete t-sequential history \( S^i \) as follows:

- for every transaction \( T_k \) that is t-complete in \( H^i \), \( S^i[k] = S[k] \).
• for every transaction $T_k$ that is complete but not t-complete in $H^i$, $S^i|k$ consists of the sequence of events in $H^i|k$, immediately followed by $tryC_k() \cdot A_k$.

• for every transaction $T_k$ with an incomplete t-operation $op_k = read_k \lor write_k \lor tryA_k()$ in $H^i$, $S^i|k$ is the sequence of events in $S|k$ up to the invocation of $op_k$, immediately followed by $A_k$.

• for every transaction $T_k \in txns(H^i)$ with an incomplete t-operation $op_k = tryC_k()$, $S^i|k = S|k$.

By the above construction, $S^i$ is indeed a t-complete history and every transaction that appears in $S^i$ also appears in $S$. Now we order transactions in $S^i$ so that $seq(S^i)$ is a subsequence of $seq(S)$.

Note that $S^i$ is derived from events contained in some completion $\bar{H}$ of $H$ that is equivalent to $S$. Since $S^i$ contains events from every complete t-operation in $H^i$ and other events included are borrowed from $\bar{H}$, there exists a completion of $H^i$ that is equivalent to $S^i$.

We now prove that $S^i$ is a serialization of $H^i$. First we observe that $S^i$ respects the real-time order of $H^i$. Indeed, if $T_j \prec^R T^i_H T_k$, then $T_j \prec^R T^i_H T_k$ and $T_j <_S T_k$. Since $seq(S^i)$ is a subsequence of $seq(S)$, we have $T_j <_S T_k$.

To show that $S^i$ is legal, suppose, by way of contradiction, that there is some $read_k(X)$ that returns $v \neq A_k$ in $H^i$ such that $v$ is not the latest written value of $X$ in $S^i$. If $T_k$ contains a $write_k(X, v')$ preceding $read_k(X)$ such that $v \neq v'$ and $v$ is not the latest written value for $read_k(X)$ in $S^i$, it is also not the latest written value for $read_k(X)$ in $S$, which is a contradiction. Thus, the only case to consider is when $read_k(X)$ should return a value written by another transaction.

Since $S$ is a serialization of $H$, there exists a committed transaction $T_m$ that performs the last $write_m(X, v)$ that precedes $read_k(X)$ in $T_k$ in $S$. Moreover, since $read_k(X)$ is legal in the local serialization for $read_k(X)$ in $H$ with respect to $S$, the prefix of $H$ up to the response of $read_k(X)$ must contain an invocation of $tryC_m()$. Thus, $read_k(X) \prec^R T_{\bar{H}}^{RT} tryC_m()$ and $T_m \in txns(H^i)$. By construction of $S^i$, $T_m \in txns(S^i)$ and $T_m$ is committed in $S^i$.

We have assumed, towards a contradiction, that $v$ is not the latest written value for $read_k(X)$ in $S^i$. Hence, there exists a committed transaction $T_j$ that performs $write_j(X, v')$ such that $T_m <_S T_j <_S T_k$. But this is not possible since $seq(S^i)$ is a subsequence of $seq(S)$.

Thus, $S^i$ is a legal t-complete t-sequential history equivalent to some completion of $H^i$. Now, by the construction of $S^i$, for every $read_k(X)$ that does not return $A_k$ in $S^i$, we have $S^i|H^i = S^i_k X$. Indeed, the transactions that appear before $T_k$ in $S^i_k X$ are those with a $tryC$ event before the response of $read_k(X)$ in $H$ and are committed in $S$. Since $seq(S^i)$ is a subsequence of $seq(S)$, we have $S^i_k X \prec^R_{\bar{H}} S^i_k X$. Thus, $read_k(X)$ is legal in $S^i_k X$.

Lemma 1 implies that every prefix of a du-opaque history has a du-opaque serialization and thus:

**Corollary 2.** DU-Opacity is a prefix-closed property.

We show, however, that du-opacity is, in general, not limit-closed. We present an infinite history that is not du-opaque, but every its prefix is.

**Proposition 1.** DU-Opacity is not a limit-closed property.

**Proof.** Let $H^j$ denote a finite prefix of $H$ of length $j$. Consider an infinite history $H$ that is the limit of the histories $H^j$ defined as follows (see Figure 2):

\[ H = H^1 \cup \cdots \cup H^j \cup \cdots \]

where $H^j$ is a finite prefix of $H$ of length $j$. Since $H^j$ is finite, it is not du-opaque. However, every prefix of $H$ is du-opaque, as shown in the proof of Lemma 1.
– Transaction \( T_1 \) performs a \( \text{write}_1(X, 1) \) and then invokes \( \text{try}C_1() \) that is incomplete in \( H \).

– Transaction \( T_2 \) performs a \( \text{read}_2(X) \) that overlaps with \( \text{try}C_1() \) and returns 1.

– There are infinitely many transactions \( T_i, i \geq 3 \), each of which performing a single \( \text{read}_i(X) \) that returns 0 such that each \( T_i \) overlaps with both \( T_1 \) and \( T_2 \).

A \( \ell \)-complete \( \ell \)-sequential history \( S^j \) is derived from the sequence \( T_3, \ldots, T_j, T_0, T_1 \) in which

1. \( \text{try}C_1() \) is completed by inserting \( C_1 \) immediately after its invocation and
2. any incomplete \( \text{read}_j(X) \) is completed by inserting \( A_j \) immediately after its invocation. It is easy to observe that \( S^j \) is indeed a serialization of \( H^j \).

However, there is no serialization of \( H \). Suppose that such a serialization \( S \) exists. Since every transaction that participates in \( H \) must participate in \( S \), there exists \( n \in \mathbb{N} \) such that \( \text{seq}(S)[n] = T_1 \). Consider the transaction at index \( n + 1 \), say \( T_i \) in \( \text{seq}(S) \). But for any \( i \geq 3 \), \( T_i \) must precede \( T_1 \) in any serialization (by legality), which is a contradiction.

We next prove that du-opacity is limit-closed if we assume that, in an infinite history, every transaction eventually completes (but not necessarily \( \ell \)-complete).

The proof uses König’s Path Lemma on a rooted directed graph, \( G \). Let \( v_0 \) be the root vertex of \( G \). We say that \( v_k \), a vertex of \( G \), is reachable from \( v_0 \), if there is a sequence of vertices \( v_0, \ldots, v_k \) such that for each \( i \), there exists an edge from \( v_i \) to \( v_{i+1} \). \( G \) is connected if every vertex in \( G \) is reachable from \( v_0 \). \( G \) is finitely branching if every vertex in \( G \) has a finite out-degree. \( G \) is infinite if the set of vertices in \( G \) is not finite.

Lemma 3 (König’s Path Lemma \([13]\)). If \( G \) is an infinite connected finitely branching rooted directed graph, then \( G \) contains an infinite sequence of vertices \( v_0, v_1, \ldots \) such that \( v_0 \) is the root, for every \( i \geq 0 \), there is an edge from \( v_i \) to \( v_{i+1} \), and for every \( i \neq j \), \( v_i \neq v_j \).

We first prove the following lemma concerning du-opaque serializations.

For a transaction \( T \in \text{txns}(H) \), we define the live set of \( T \) in \( H \), denoted \( \text{Lset}_H(T) \) (\( T \) included) as follows: every transaction \( T' \in \text{txns}(H) \) such that neither the last event of \( T' \) precedes the first event of \( T \) in \( H \) nor the last event of \( T \) precedes the first event of \( T' \) in \( H \) is contained in \( \text{Lset}_H(T) \). We say that transaction \( T' \in \text{txns}(H) \) succeeds the live set of \( T \) and we write \( T \prec_{\text{L}}^H T' \) if in \( H \), for all \( T'' \in \text{Lset}_H(T) \), \( T'' \) is complete and the last event of \( T'' \) precedes the first event of \( T' \).

Lemma 4. Let \( H \) be a finite du-opaque history and assume \( T_k \in \text{txns}(H) \) be a complete transaction in \( H \) such that every transaction in \( \text{Lset}_H(T_k) \) is complete in \( H \). Then there exists a serialization \( S \) of \( H \) such that for all \( T_k, T_m \in \text{txns}(H) \); \( T_k \prec_{\text{L}}^H T_m \), we have \( T_k <_S T_m \).

Proof. Since \( H \) is du-opaque, there exists a serialization \( \tilde{S} \) of \( H \).

Let \( S \) be a \( \ell \)-complete \( \ell \)-sequential history such that \( \text{txns}(\tilde{S}) = \text{txns}(S) \), and \( \forall T_i \in \text{txns}(\tilde{S}) : S|i = \tilde{S}|i \). We now perform the following procedure iteratively to derive \( \text{seq}(S) \) from \( \text{seq}(\tilde{S}) \).

Initially \( \text{seq}(S) = \text{seq}(\tilde{S}) \). For each \( T_k \in \text{txns}(H) \), let \( T_k \in \text{txns}(H) \) denote the earliest transaction in \( \tilde{S} \) such that \( T_k \prec_{\text{L}}^H T_k \) and \( T_k <_S T_k \) (implying \( T_k \) is not \( \ell \)-complete), then move \( T_k \) to immediately precede \( T_k \) in \( \text{seq}(S) \).

By construction, \( S \) is equivalent to \( \tilde{S} \) and for all \( T_k, T_m \in \text{txns}(H) ; T_k \prec_{\text{L}}^H T_m, T_k <_S T_m \). We claim that \( S \) is a serialization of \( H \). Observe that any two transactions that are complete in \( H \), but not \( \ell \)-complete are not related by real-time order in \( H \). By construction of \( S \), for any transaction \( T_k \in \text{txns}(H) \), the set of transactions that precede \( T_k \) in \( \tilde{S} \), but succeed \( T_k \) in \( S \) are not related to \( T_k \) by real-time order. Since \( \tilde{S} \) respects the real-time order in \( H \), this holds also for \( S \).
We now show that $S$ is legal. Consider any \texttt{read}_k(X) performed by some transaction $T_k$ that returns $v \in V$ in $S$ and let $T_i \in \text{txns}(H)$ be the earliest transaction in $\bar{S}$ such that $T_k \prec_H^S T_i$. Suppose, by contradiction, that \texttt{read}_k(X) is not legal in $S$. Thus, there exists a committed transaction $T_m$ that performs \texttt{write}_m(X, v) in $S$ such that $T_m = T_i$ or $T_i <_S T_m <_S T_k$. Note that, by our assumption, \texttt{read}_k(X) \not\preceq^H_{RT} \texttt{tryC}_i()$. Since \texttt{read}_k(X) must be legal in the local serialization of $\bar{S}$ with respect to $H$, \texttt{read}_k(X) \not\preceq^H_{RT} \texttt{tryC}_m()$. Thus, $T_m \in L\text{set}_H(T_k)$. Therefore $T_m \neq T_i$. Moreover, $T_m$ is complete, and since it commits in $\bar{S}$, it is also t-complete in $H$ and the last event of $T_m$ precedes the first event of $T_i$ in $H$, i.e., $T_m \prec^H_{RT} T_i$. Hence, $T_i$ cannot precede $T_m$ in $\bar{S}$—a contradiction.

Observe also that since $T_k$ is complete in $H$ but not t-complete, $H$ does not contain an invocation of \texttt{tryC}_k(). Thus, the legality of any other transaction is unaffected by moving $T_k$ to precede $T_i$ in $S$. Thus, $S$ is a legal t-complete t-sequential history equivalent to some completion of $H$. The above arguments also prove that every t-read in $S$ is legal in its local serialization with respect to $H$ and $S$ and, thus, $S$ is a serialization of $H$.

\begin{theorem}
Under the restriction that in any infinite history $H$, every transaction $T_k \in \text{txns}(H)$ is complete, du-opacity is a limit-closed property.
\end{theorem}

\begin{proof}
We are given an infinite sequence of finite ever-extending du-opaque histories, let $H$ be the corresponding infinite limit history. We want to show that $H$ is also du-opaque. By Corollary 2 every prefix of $H$ is du-opaque. Therefore, we can assume the sequence of du-opaque histories to be $H^0, H^1, \ldots, H^i, H^{i+1}, \ldots$, where each $H^i$ is the prefix of $H$ of length $i$.

We construct a rooted directed graph $G_H$ as follows:

(0) The root vertex of $G_H$ is $(H^0, S^0)$ where $S^0$ and $H^0$ contain the initial transaction $T_0$.

(1) Each non-root vertex of $G_H$ is a tuple $(H^i, S^i)$, where $S^i$ is a du-opaque serialization of $H^i$ that satisfies the condition specified in Lemma 4 for all $T_k, T_m \in \text{txns}(H)$; $T_k \prec^S_H T_m, T_k <_S T_m$. Note that there exist several possible serializations for any $H^i$. For succinctness, in the rest of this proof, when we refer to a specific $S^i$, it is understood to be associated with the prefix $H^i$ of $H$.

(2) We say that a transaction $T$ is \textit{complete} in $H'$ with respect to $H$, where $H$ is any extension of $H'$ if last step of $T$ in $H$ is a response event and it is contained in $H'$.

Let $cseq_k(S^i), j \geq i$, denote the subsequence of seq$(S^i)$ reduced to transactions that are complete in $H^i$ with respect to $H$. For every pair of vertices $v = (H^i, S^i)$ and $v' = (H^{i+1}, S^{i+1})$ in $G_H$, there is an edge from $v$ to $v'$ if $cseq_k(S^i) = cseq_k(S^{i+1})$.

The out-degree of a vertex $v = (H^i, S^i)$ in $G_H$ is defined by the number of possible serializations of $H^{i+1}$, bounded by the number of possible permutations of the set $\text{txns}(S^{i+1})$, implying that $G_H$ is finitely branching.

By Lemma 1, given any serialization $S^{i+1}$ of $H^{i+1}$, there exists a serialization $S^i$ of $H^i$ such that seq$(S^i)$ is a subsequence of seq$(S^{i+1})$. Indeed, the serialization $S^i$ of $H^i$ also respects the restriction specified in Lemma 4. Since seq$(S^{i+1})$ contains every complete transaction that takes its last step in $H$ in $H^i$, $cseq_k(S^i) = cseq_k(S^{i+1})$. Therefore, for every vertex $(H^{i+1}, S^{i+1})$, there is a vertex $(H^i, S^i)$ such that $cseq_k(S^i) = cseq_k(S^{i+1})$. Thus, we can iteratively construct a path from $(H^0, S^0)$ to every vertex $(H^i, S^i)$ in $G_H$, implying that $G_H$ is connected.

We now apply König’s Path Lemma to $G_H$. Since $G_H$ is an infinite connected finitely branching rooted directed graph, we can derive an infinite sequence of non-repeating vertices

$L = (H^0, S^0), (H^1, S^1), \ldots, (H^i, S^i), \ldots$.
Claim 8. Let $H$ be a subsequence of $H^3$ reduced to transactions in $S$ such that each $T_k \in \text{trans}(S)$ is complete in $H^3$ with respect to $H$. Then, for every $i$, there is $j$ such that $S$ is a serialization of $H^3_i$.
tryC
→
C
R
(X) → 1
W (X, 1)
T
1

Figure 3: History H is final-state opaque, while its prefix H' is not final-state opaque.

Proof. Let H^j be the shortest prefix of H (from L) such that for each T ∈ txns(\hat{S}^i), if seq(S^j)[k] = T, then for every j’ > j, seq(S^j')[k] = T. From the construction of F, such j and k exist. Also, we observe that txns(\hat{S}^i) ⊆ txns(S^j) and F^i is a subsequence of seq(S^j). Using arguments similar to the proof of Lemma 1, it follows that \hat{S}^i is indeed a serialization of H^j.

Since H is complete, there is exactly one completion of H, where each transaction T_k that is not t-complete in H is completed with tryC_k · A_k after its last event. By Claim 8, the limit t-sequential t-complete history is equivalent to this completion, is legal, respects the real-time order of H, and ensures that every read is legal in the corresponding local serialization. Thus, S is a serialization of H.

From Theorem 5 it follows that:

**Corollary 9.** Let M be any TM implementation that ensures that in every infinite history H of M, each transaction T ∈ txns(H) is complete in H. Then, M is du-opaque iff every finite history of M is du-opaque.

4 Comparison with Other TM Consistency Definitions

4.1 Relation to Opacity

In this section, we relate du-opacity with opacity, as defined by Guerraoui and Kapalka [8]. Note that the definition presented in [8] applies to any object with a sequential specification. For the sake of comparison, we restrict it here to TMs with read-write semantics.

**Definition 4** (Guerraoui and Kapalka [7, 8]). A finite history H is final-state opaque if there is a legal t-complete t-sequential history S, such that

1. for any two transactions T_k, T_m ∈ txns(H), if T_k ≺^R^T H T_m, then T_k ≺_S T_m, and
2. S is equivalent to a completion of H (cf. Definition 2).

We say that S is a final-state serialization of H.

Figure 3 presents a t-complete sequential history H, demonstrating that final-state opacity is not a prefix-closed property. H is final-state opaque, with T_1 · T_2 being a legal t-complete t-sequential history equivalent to H. Let H’ = write_1(X, 1), read_2(X) be a prefix of H in which T_1 and T_2 are t-incomplete. By Definition 2 T_i (i = 1, 2) is completed by inserting tryC_i · A_i immediately after the last event of T_i in H. Observe that neither T_1 · T_2 nor T_2 · T_1 are sequences that allow us to derive a serialization of H’ (we assume that the initial value of X is 0).

A restriction of final-state opacity, which we refer to as opacity, was presented in [8] by filtering out histories that are not prefix-closed.

10
Definition 5 (Guerraoui and Kapalka [8]). A history $H$ is opaque if and only if every finite prefix $H'$ of $H$ (including $H$ itself if it is finite) is final-state opaque.

It can be easily seen that opacity is prefix and limit-closed, and, thus, opacity is a safety property.

Proposition 2. There is an opaque history that is not du-opaque.

Proof. Consider the finite history $H$ depicted in Figure 4. To prove that $H$ is opaque, we proceed by examining every prefix of $H$.

1. Each prefix up to the invocation of $\text{read}_2(X)$ is trivially final-state opaque.

2. Consider the prefix, $H^i$ of $H$ where the $i^{th}$ event is the response of $\text{read}_2(X)$. Let $S^i$ be a t-complete t-sequential history derived from the sequence $T_1, T_2$ by inserting $C_1$ immediately after the invocation of $\text{try}C_1()$. It is easy to see that $S^i$ is a final-state serialization of $H^i$.

3. Consider the t-complete t-sequential history $S$ derived from the sequence $T_1, T_3, T_2$ in which each transaction is t-complete in $H$. Clearly, $S$ is a final-state serialization of $H$.

Since $H$ and every (proper) prefix of it are final-state opaque, $H$ is opaque.

Clearly, the only final-state serialization $S$ of $H$ is specified by $\text{seq}(S) = T_1, T_3, T_2$. Consider $\text{read}_2(X)$ in $S$: since $H^{2,X}$, the prefix of $H$ up to the response of $\text{read}_2(X)$ does not contain an invocation of $\text{try}C_3()$, the local serialization for $\text{read}_2(X)$ with respect to $H$ and $S$, $S^{2,X}_H$ is $T_1 \cdot \text{read}_2(X)$. But $\text{read}_2(X)$ is not legal in $S^{2,X}_H$ — contradiction. Thus, $H$ is not du-opaque.

Theorem 10. DU-Opacity $\subset$ Opacity.

Proof. We first claim that every finite du-opaque history is opaque. Let $H$ be a finite du-opaque history. By definition, there exists a final-state serialization $S$ of $H$. Since du-opacity is a prefix-closed property, every prefix of $H$ is final-state opaque. Thus, $H$ is opaque.

Again, since every prefix of a du-opaque history is also du-opaque, by Definition 5 every infinite du-opaque history is also opaque.

Proposition 2 now establishes that du-opacity is indeed a restriction of opacity.

We now show that du-opacity is equivalent to opacity assuming that no two transactions write identical values to the same t-object (“unique-write” assumption).

Let $\text{Opacity}_{ut} \subseteq \text{Opacity}$, be a property defined as follows:

1. an infinite opaque history $H \in \text{Opacity}_{ut}$ iff every transaction $T \in \text{txns}(H)$ is complete in $H$, and

2. an opaque history $H \in \text{Opacity}_{ut}$ iff for any two transactions $T_k, T_m \in \text{txns}(H)$ that perform $\text{write}_k(X, v)$ and $\text{write}_m(X, v')$ respectively, $v \neq v'$.

Figure 4: History is opaque, but not du-opaque
tryC opaque. Theorem 5 now implies that Opacity is not legal in Figure 5, which is du-opaque. Thus, there exists a different completion of the transaction that does not contain any invocation of tryC(). Since read_k(X) is legal in S, there exists a committed transaction \( T_m \in \text{txns}(H) \) that performs write_m(X, v) that is the latest such write in S that precedes \( T_k \). Thus, if read_k(X) is not legal in \( S^k,X_H \), the only possibility is that read_k(X) \( \not\in \) H tryC_m(). Under the assumption of unique writes, there does not exist any other transaction \( T_j \in \text{txns}(H) \) that performs write_j(X, v). Consequently, there does not exist any \( \hat{H}^k,X \) (some completion of \( H^k,X \)) and t-complete t-sequential history \( S' \) such that \( S' \) is equivalent to \( H^k,X \) and \( S' \) contains any committed transaction that writes v to X i.e. \( H^k,X \) is not final-state opaque. However, since \( H \) is opaque, every prefix of \( H \) must be final-state opaque—contradiction.

By Definition 5, an infinite history \( H \) is opaque if every finite prefix of \( H \) is final-state opaque. Theorem 4 now implies that Opacity_{ut} \( \subseteq \) DU-Opacity.

By Definition 5 and Corollary 2, it follows that DU-Opacity \( \subseteq \) Opacity_{ut}.

4.2 Relation with Other definitions

Explicitly using the deferred-update semantics in an opacity definition was first proposed by Guerraoui et al. [6] and later adopted by Kuznetsov and Ravi [14]. In both papers, opacity is only defined on sequential histories, where every invocation of a t-operation is immediately followed by a matching response. In particular, these definitions require the final-state serialization to respect the read-commit order: \( H \) is opaque by their definition if there exists a final-state serialization \( S \) of \( H \) such that if a t-read of a t-object \( X \) by a transaction \( T_k \) precedes the tryC of a transaction \( T_m \) that commits on \( X \) in \( H \), then \( T_k \) precedes \( T_m \) in \( S \). But we observe that this definition is not equivalent to opacity even for sequential histories. In fact the property defined in [6] is strictly stronger than du-opacity: the sequential history in Figure 5 is du-opaque.
(and consequently opaque by Theorem 10). We can derive a du-opaque serialization $S$ for this history such that $\text{seq}(S) = T_1, T_3, T_2$. In fact, this is the only final-state serialization for $H$. However, by the above definition, $T_2$ must precede $T_3$ in any serialization of this history since the response of $\text{read}_2(X)$ precedes the invocation of $\text{tryC}_3()$. Thus, $H$ is not opaque by the definition in [6].

The recently introduced TMS2 correctness condition [5, 15] is another attempt to clarify opacity. Two transactions are said to conflict in a given history if they access the same t-object and at least one of them successfully commits to it. Informally, for each history $H$ in TMS2, there exists a final-state serialization $S$ of $H$ such that if two transactions $T_1$ and $T_2$ conflict on t-object $X$ in $H$, where $X \in Wset(T_1) \cap Rset(T_2)$ and $\text{tryC}$ of $T_1$ precedes the $\text{tryC}$ of $T_2$, then $T_1$ must precede $T_2$ in $S$. We conjecture that every history in TMS2 is du-opaque, but not vice-versa. Figure 6 depicts a history $H$ that is du-opaque, but not TMS2. Indeed, there exists a du-opaque serialization $S$ of $H$ such that $\text{seq}(S) = T_2, T_1$. On the other hand, $T_1$ and $T_2$ are in conflict, $T_1$ commits before $T_2$, but there does not exist any final-state serialization of $H$ in which $T_1$ precedes $T_2$.

5 Discussion

It is widely accepted that a correctness condition on a set of histories should be a safety property, i.e., should be prefix- and limit-closed. The definition of opacity proposed in [8] forcefully achieves prefix-closure by restricting final-state opacity to prefix-closed histories, and trivially achieves limit-closure by reducing correctness of an infinite history to correctness of its prefixes.

This paper proposes a correctness criterion that explicitly disallows reading from an uncommitted transaction, which ensures prefix-closure and (under the restriction that every transaction eventually completes every operation it invokes, but not necessarily commits or aborts) limit-closure. We believe that this constructive definition is useful to TM practitioners, since it streamlines possible implementations of t-read and $\text{tryC}$ operations. Moreover, it seems that du-opacity already captures the sets of histories exported by most existing opaque TM implementations [3, 4, 10]. In contrast, the recent pessimistic STM implementation [1], in which no transaction aborts, does not intend to provide the deferred-update semantics and, thus, is not in the focus of this paper. Technically, the pessimistic STM of [1] is not opaque, and certainly, not du-opaque.

To the best of our knowledge, there is no prior work proving that any TM correctness property is a safety property in the formal sense. The argumentation in the proof of Theorem 5 is inspired by the proof sketch in [16] of the safety of linearizability [12], but turns out to be trickier due to the more complicated definition of du-opacity.

Acknowledgements

The authors are grateful to the anonymous reviewers for insightful comments on the previous versions of this paper, and Victor Luchangco for interesting discussions.

This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement Number 238639, ITN project TRANSFORM, and grant agreement Number 248465, the S(o)OS project.
References

[1] Yehuda Afek, Alexander Matveev, and Nir Shavit. Pessimistic software lock-elision. In DISC, pages 297–311, 2012.

[2] Bowen Alpern and Fred B. Schneider. Defining liveness. Information Processing Letters, 21(4):181–185, October 1985.

[3] Luke Dalessandro, Michael F. Spear, and Michael L. Scott. Norec: streamlining stm by abolishing ownership records. In Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP ’10, pages 67–78, New York, NY, USA, 2010. ACM.

[4] Dave Dice, Ori Shalev, and Nir Shavit. Transactional locking ii. In In Proc. of the 20th Intl. Symp. on Distributed Computing, 2006.

[5] Simon Doherty, Lindsay Groves, Victor Luchangco, and Mark Moir. Towards formally specifying and verifying transactional memory. Electron. Notes Theor. Comput. Sci., 259:245–261, December 2009.

[6] Rachid Guerraoui, Thomas A. Henzinger, and Vasu Singh. Permissiveness in transactional memories. In DISC, pages 305–319, 2008.

[7] Rachid Guerraoui and Michal Kapalka. On the correctness of transactional memory. In PPOPP, pages 175–184, 2008.

[8] Rachid Guerraoui and Michal Kapalka. Principles of Transactional Memory, Synthesis Lectures on Distributed Computing Theory. Morgan and Claypool, 2010.

[9] Vassos Hadzilacos. A theory of reliability in database systems. J. ACM, 35(1):121–145, 1988.

[10] Maurice Herlihy, Victor Luchangco, Mark Moir, and William N. Scherer, III. Software transactional memory for dynamic-sized data structures. In Proceedings of the twenty-second annual symposium on Principles of distributed computing, PODC ’03, pages 92–101, New York, NY, USA, 2003. ACM.

[11] Maurice Herlihy and J. Eliot B. Moss. Transactional memory: architectural support for lock-free data structures. SIGARCH Comput. Archit. News, 21(2):289–300, 1993.

[12] Maurice Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst., 12(3):463–492, 1990.

[13] Dénes König. Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe. Akad. Verlag. 1936.

[14] Petr Kuznetsov and Srivatsan Ravi. On the cost of concurrency in transactional memory. CoRR, abs/1103.1302, 2011.

[15] Mohsen Lesani, Victor Luchangco, and Mark Moir. Putting opacity in its place. In WTTM, 2012.

[16] Nancy A. Lynch. Distributed Algorithms. Morgan Kaufmann, 1996.

[17] Susan S. Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. ACM Trans. Program. Lang. Syst., 4(3):455–495, 1982.
[18] Nir Shavit and Dan Touitou. Software transactional memory. In *PODC ’95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, pages 204–213, 1995.