Safety and Deferred Update in Transactional Memory

Hagit Attiya[1], Sandeep Hans[1], Petr Kuznetsov[2], and Srivatsan Ravi[3]

[1] Technion
{hagit,sandeep}@cs.technion.ac.il
[2] Télécom ParisTech
petr.kuznetsov@telecom-paristech.fr
[3] TU Berlin
srivatsan@srivatsan.in

Abstract. Transactional memory allows the user to declare sequences of instructions as speculative *transactions* that can either *commit* or *abort*, providing *all-or-nothing* semantics. If a transaction commits, it should appear to execute sequentially, so that the committed transactions constitute a correct sequential execution. If a transaction aborts, none of its instructions should affect other transactions. These semantics allow the programmer to incorporate sequential code within transactions and let the transactional memory care about conflicts between concurrent transactions. In this sense, it is important that the memory is *safe*, i.e., *every* transaction has a *consistent* view even if the transaction aborts later. Otherwise, inconsistencies not predicted by the sequential program may cause a fatal irrecoverable error or an infinite loop. Furthermore, in a general setting, where a transaction may be explicitly aborted by the user or an external contention manager, a transaction should not be allowed to read from a not yet committed transaction, which is often called *deferred-update* semantics. This chapter overviews the scope of consistency criteria proposed so far to capture deferred-update semantics, and shows that—under reasonable conditions—the semantics induces a safety property.

1 Introduction

Resolving conflicts in an efficient and consistent manner is a big challenge in concurrent software design. *Transactional memory* (TM) [10, 19] 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 *atomic*: 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 TM take care of *conflicts* (concurrent reading and writing to the same memory location) resulting from concurrent executions.

In databases, a correct implementation of concurrency control should guarantee that committed transactions constitute a serial execution [9]. Uncommitted transactions can be aborted without invalidating the correctness of committed ones. (In the literature on databases, the latter feature is called *recoverability* [9].)

© Springer International Publishing Switzerland 2015

In the TM context, intermediate states witnessed by the read operations of an incomplete transaction may affect the application. If the intermediate state is not consistent with any sequential execution, the application may experience a fatal irrecoverable error or enter an infinite loop. Thus, it is important that *each* transaction, including *aborted* ones observes a *consistent* state.

A state should be considered consistent if it could result from a serial application of transactions observed in the current execution. In this sense, every transaction should witness a state that *could have been* observed in *some* execution of the sequential code put by the programmer within the transactions. Additionally, a consistent state should not depend on a transaction that has not started committing yet (referred to as *deferred-update* semantics). This restriction appears desirable, since the ongoing transaction may still abort (explicitly by the user or because of consistency reasons) and, thus, render the read inconsistent. Further, the set of histories specified by the consistency criterion must constitute a *safety property*, as defined by Owicki and Lamport [17], Alpern and Schneider [1] and refined by Lynch [16]: it must be non-empty, *prefix-closed* and *limit-closed.*

In this chapter, we define the notion of deferred-update semantics formally, which we then apply to a spectrum of TM consistency criteria. Additionally, we verify if the resulting TM consistency criterion is a safety property, as defined by Lynch [16].

We first consider the popular criterion of *opacity* [7]. Opacity requires the states observed by all transactions, included uncommitted ones, to be consistent with a global *serialization*, i.e., a serial execution constituted by committed transactions. Moreover, the serialization should respect the *real-time order*: a transaction that completed before (in real time) another transaction started should appear first in the serialization.

By definition, opacity reduces correctness of a history to correctness of all its prefixes, and thus is prefix-closed and limit-closed by definition. Thus, to verify that a history is opaque, one needs to verify that each of its prefixes is consistent with some global serialization. To simplify verification and explicitly introduce deferred-update semantics into a TM correctness criterion, we specify a general criterion of *du-opacity* [3], which requires the global serial execution to respect the deferred-update property. Informally, a du-opaque history must be indistinguishable from a totally-ordered history, with respect to which no transaction reads from a transaction that has not started committing.

Du-opacity is *prefix-closed*, that is, every prefix of a du-opaque history is also du-opaque. We then show that extending opacity (and du-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. However, under certain restrictions, we show that du-opacity is *limit-closed.* In particular, assuming that in an infinite history, every transaction completes each of the operations it invoked, the limit of any sequence of ever extending du-opaque histories is also du-opaque. Therefore, under this assumption, du-opacity is a *safety property* [1, 16, 17], and to prove that a TM implementation that complies with the assumption is du-opaque, it suffices to prove that all its *finite* histories are du-opaque.

One may notice that the intended safety semantics does not require that all transactions observe the same serial execution. Intuitively, we only need that every transaction witnesses *some* consistent state, while the views of different aborted transactions do not have to be consistent with *the same* serial execution. As long as committed transactions constitute a serial execution and every transaction witnesses a consistent state, the execution can be considered "safe": no run-time error that cannot occur in a serial execution can happen. Recently, several definitions adopted this approach: *virtual-world consistency (VWC)* [12] and *Transactional Memory Specifications (TMS)* [5]. We introduce "deferred-update" versions of these proporties and discuss how the resulting properties relate to du-opacity.

The chapter is organized as follows. In Section 2, we introduce our model definitions, recall the notion of safety, and recall the original definition of opacity. In Section 3, we define du-opacity and discuss the property from the safety perspective. In Section 4, we relate du-opacity to the conventional notion of opacity [7]. In Section 5, we compare du-opacity to other TM correctness criteria, such as VWC [12], TMS1 and TMS2 [5], restricted to provide the deferred-update semantics. Section 6 gives a summary of our comparative analysis and concludes the chapter.

2 Preliminaries

A *transactional memory* (in short, *TM*) supports atomic *transactions*. Each transaction is a sequence of accesses, reading from and writing to a set of *transactional* objects (in short, *t-objects*). Each transaction T_k has a unique identifier k.

A transaction T_k accesses t-objects with *t-operations*, each being a matching pair of *invocation* and *response* events: $read_k(X)$ returns a value in some domain V or a special value $A_k \notin V$ (*abort*); $write_k(X,v)$, for a value $v \in V$, returns ok_k or A_k; $tryA_k$ returns A_k; $tryC_k$ returns a special value $C_k \notin V \cup \{A_k\}$ (*commit*) or A_k.

2.1 Implementations and Histories

We consider an asynchronous shared-memory system in which processes communicate via transactions. A TM *implementation* provides processes with algorithms for implementing $read_k$, $write_k$, $tryC_k()$ and $tryA_k()$ 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 for 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, $H|k$ begins with an invocation of a t-operation, $H|k$ is sequential and has no events after A_k or C_k. For simplicity, 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 op_k after receiving C_k or A_k. Note that the assumption excludes the TM designs providing *nested parallelism* discussed in a dedicated chapter of this book.

The *read set* of a transaction T_k in history H, denoted $Rset(T_k)$, is the set of t-objects that T_k reads in H; the *write set* of T_k in history H, denoted $Wset(T_k)$, is the set of t-objects T_k writes to in H. More specifically, we say that $X \in Rset(T_k)$ (resp., $X \in Wset(T_k)$) in H if H contains an invocation of $read_k(X)$ (resp., $write_k(X,v)$). We avoid parameterizing $Rset(T_k)$ and $Wset(T_k)$ with the history H since it is clear from the usage. If $Wset(T_k) \neq \emptyset$, then T_k is an *updating* transaction.

2.2 Complete Histories and Real-Time Precedence

A transaction $T_k \in txns(H)$ is *complete in* a history H if $H|k$ ends with a response event. A 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 op_k, op_j, we say that op_k *precedes* op_j in the *real-time order* of H, denoted $op_k \prec_H^{RT} op_m$, if the response of op_k precedes the invocation of op_j.

We overload the notation and say, for transactions $T_k, T_m \in txns(H)$, that T_k *precedes* T_m in the *real-time order* of H, denoted $T_k \prec_H^{RT} T_m$, 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 $T_k \prec_H^{RT} T_m$ nor $T_m \prec_H^{RT} T_k$, 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” t-complete transaction T_0 that writes initial values to all t-objects and commits before any other transaction begins in H.

2.3 Latest Written Value and Legality

Let H be a t-sequential history. For every operation $read_k(X)$ in H, we define the *latest written value* of X as follows: if T_k contains a $write_k(X,v)$ preceding $read_k(X)$, then the latest written value of X is the value of the latest such write to X. Otherwise, the latest written value of X is the value of the argument of the latest $write_m(X,v)$ that precedes $read_k(X)$ and belongs to a committed transaction in H. (This write is well-defined since H starts with T_0 writing to all t-objects.)

We say that $read_k(X)$ is *legal* in a t-sequential history H if it returns the latest written value of X, and H is *legal* if every $read_k(X)$ in H that does not return A_k is legal in H.

We also assume, for simplicity, that the client invokes a $read_k(X)$ 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 legality.

2.4 Safety

A *property* $\mathscr{P}$ is a set of (transactional) histories.

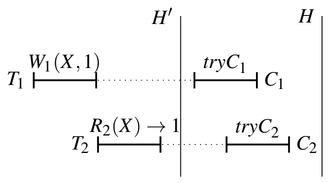

Fig. 1. History H is final-state opaque, while its prefix H' is not final-state opaque

Definition 1 (Lynch [16]). *A property $\mathcal{P}$ is a* safety *property if it satisfies the following two conditions:*

Prefix-closure: *For every history $H \in \mathcal{P}$, every prefix H' of H (i.e., every prefix of the sequence of the events in H) is also in $\mathcal{P}$.*

Limit-closure: *For every infinite sequence of finite histories $H^0, H^1, \ldots$ such that for every i, $H^i \in \mathcal{P}$ and H^i is a prefix of H^{i+1}, the limit of the sequence is also in $\mathcal{P}$.*

Notice that the set of histories produced by a TM implementation M is, by construction, 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.

2.5 Opacity

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

- *First, for every transaction $T_k \in txns(H)$ with an incomplete t-operation op_k in H, if $op_k = read_k \vee write_k$, insert A_k somewhere after the invocation of op_k; otherwise, if $op_k = tryC_k()$, insert C_k or A_k somewhere after the last event of T_k.*
- *After all transactions are complete, for every transaction T_k that is not t-complete, insert $tryC_k \cdot A_k$ after the last event of transaction T_k.*

Definition 3 (Guerraoui and Kapalka [7]). *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 $T_k, T_m \in txns(H)$, if $T_k \prec_H^{RT} T_m$, then $T_k <_S T_m$, and*
2. *S is equivalent to a completion of H.*

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

Final-state opacity is not prefix-closed. Figure 1 depicts a t-complete sequential history H that is final-state opaque, with $T_1 \cdot T_2$ 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. Transaction T_i ($i = 1,2$) is completed by inserting $tryC_i \cdot A_i$ immediately after the last event of T_i in H. Observe that neither $T_1 \cdot T_2$ nor $T_2 \cdot T_1$ 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* [7] explicitly filters out histories that are not prefix-closed.

Definition 4 (Guerraoui and Kapalka [7]). *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, it is a safety property.

3 Deferred-Update Semantics and Its Properties

We now give a formal definition of opacity with deferred-update semantics. Then we show that the property is prefix-closed and, under certain *liveness* restrictions, limit-closed.

3.1 Du-Opacity

Let H be any history and let 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.

Definition 5 (Local serialization). *For any $read_k(X)$ that does not return A_k, let $S^{k,X}$ be the prefix of S up to the response of $read_k(X)$ and $H^{k,X}$ be the prefix of H up to the response of $read_k(X)$.*

$S_H^{k,X}$, the local serialization *of $read_k(X)$ with respect to H and S, is the subsequence of $S^{k,X}$ derived by removing from $S^{k,X}$ the events of all transactions $T_m \in txns(H) \setminus \{T_k\}$ such that $H^{k,X}$ does not contain an invocation of $tryC_m()$.*

We are now ready to present our correctness condition, *du-opacity*.

Definition 6. *A history H is* du-opaque *if there is a legal t-complete t-sequential history S such that*

1. *there is a completion of H that is equivalent to S, and*
2. *for every pair of transactions $T_k, T_m \in txns(H)$, if $T_k \prec_H^{RT} T_m$, then $T_k <_S T_m$, i.e., S respects the real-time ordering of transactions in H, and*
3. *each $read_k(X)$ in S that does not return A_k is legal in $S_H^{k,X}$.*

We then say that S is a (du-opaque) serialization *of H.*

Informally, a history H is du-opaque if there is 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.

For any du-opaque serialization S, $seq(S)$ denotes the *sequence of transactions* in S and $seq(S)[k]$ denotes the k^{th} transaction in this sequence.

3.2 Du-Opacity Is Prefix-Closed

Lemma 1. *Let H be a du-opaque history and let S be a serialization of H. For any $i \in \mathbb{N}$, there is a serialization S^i of H^i (the prefix of H consisting of the first i events), 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, $op_k = read_k \vee write_k \vee tryA_k()$ 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. 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 $\overline{H}$ of H that is equivalent to S and some other events to derive a completion of S^i. Since S^i contains events from every complete t-operation in H^i and other events included satisfy Definition 2, there is a completion of H^i that is equivalent to S^i.

We now argue 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 $T_j \prec_{H^i}^{RT} T_k$, then $T_j \prec_H^{RT} T_k$ and $T_j <_S T_k$. Since $seq(S^i)$ is a subsequence of $seq(S)$, we have $T_j <_{S^i} T_k$.

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 is 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 of $read_k(X)$ 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, $read_k(X) \not\prec_H^{RT} tryC_m()$ 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 is a committed transaction T_j that performs $write_j(X, v'); v' \neq v$ in S^i such that $T_m <_{S^i} T_j <_{S^i} T_k$. 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 $S^{i\,k,X}_{H^i} = S^{k,X}_H$. Indeed, the transactions that appear before T_k in $S^{i\,k,X}_{H^i}$ 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 $S^{i\,k,X}_{H^i} = S^{k,X}_H$. Thus, $read_k(X)$ is legal in $S^{i\,k,X}_{H^i}$. $\square$

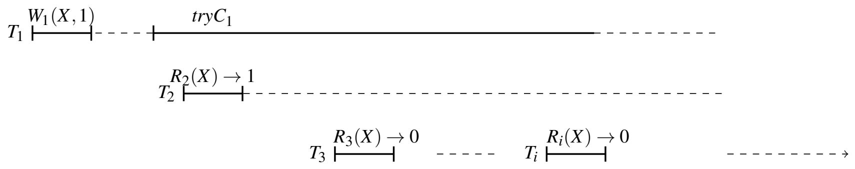

Fig. 2. An infinite history in which $tryC_1$ is incomplete and any two transactions are concurrent. Each finite prefix of the history is du-opaque, but the infinite limit of the ever-extending sequence is not du-opaque.

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

Corollary 1. *Du-opacity is a prefix-closed property.*

3.3 The Limit of Du-Opaque Histories

We observe, however, that du-opacity is, in general, not limit-closed. We present an infinite history that is not du-opaque, but each of its prefixes 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):

- Transaction T_1 performs a $write_1(X,1)$ and then invokes $tryC_1()$ that is incomplete in H.
- Transaction T_2 performs a $read_2(X)$ that overlaps with $tryC_1()$ and returns 1.
- There are infinitely many transactions T_i, $i \geq 3$, each of which performing a single $read_i(X)$ that returns 0 such that each T_i overlaps with both T_1 and T_2.

We now prove that, for all $j \in \mathbb{N}$, H^j is a du-opaque history. Clearly, H^0 and H^1 are du-opaque histories. For all $j > 1$, we first derive a completion of H^j as follows:

1. $tryC_1()$ (if it is contained in H^j) is completed by inserting C_1 immediately after its invocation,
2. for all $i \geq 2$, any incomplete $read_i(X)$ that is contained in H^j is completed by inserting A_i and $tryC_i \cdot A_i$ immediately after its invocation, and
3. for all $i \geq 2$ and every complete $read_j(X)$ that is contained in H^j, we include $tryC_i \cdot A_i$ immediately after the response of this $read_j(X)$.

We can now derive a t-complete t-sequential history S^j equivalent to the above derived completion of H^j from the sequence of transactions $T_3, \ldots, T_i, T_1, T_2$ (depending on which of these transactions participate in H^j), where $i \geq 3$. It is easy to observe that S^j so derived 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 $seq(S)[n] = T_1$. Consider the transaction at index $n+1$, say T_i in $seq(S)$. But for any $i \geq 3$, T_i must precede T_1 in any serialization (by legality), which is a contradiction. □

Notice that all finite prefixes of the infinite history depicted in Figure 2 are also opaque. Thus, if we extend the definition of opacity to cover infinite histories in a non-trivial way, i. e., by explicitly defining opaque serializations for infinite histories, we can reformulate Proposition 1 for opacity.

3.4 Du-Opacity is Limit-Closed for Complete Histories

We show now that du-opacity is limit-closed if the only infinite histories we consider are those in which every transaction eventually completes (but not necessarily t-completes).

We first prove an auxiliary lemma on du-opaque serializations. For a transaction $T \in txns(H)$, the *live set of* T *in* H, denoted $Lset_H(T)$ (T included), is defined as follows: every transaction $T' \in txns(H)$ 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 $Lset_H(T)$. We say that transaction $T' \in txns(H)$ *succeeds the live set of* T and we write $T \prec_H^{LS} T'$ if in H, for all $T'' \in Lset_H(T)$, T'' is complete and the last event of T'' precedes the first event of T'.

Lemma 2. *Let H be a finite du-opaque history and assume $T_k \in txns(H)$ is a complete transaction in H, such that every transaction in $Lset_H(T_k)$ is complete in H. Then there is a serialization S of H, such that for all $T_k, T_m \in txns(H)$, if $T_k \prec_H^{LS} T_m$, then $T_k <_S T_m$.*

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

Let S be a t-complete t-sequential history such that $txns(\tilde{S}) = txns(S)$, and $\forall\ T_i \in txns(\tilde{S}) : S|i = \tilde{S}|i$. We now perform the following procedure iteratively to derive $seq(S)$ from $seq(\tilde{S})$. Initially $seq(S) = seq(\tilde{S})$. For each $T_k \in txns(H)$, let $T_\ell \in txns(H)$ denote the earliest transaction in $\tilde{S}$ such that $T_k \prec_H^{LS} T_\ell$. If $T_\ell <_{\tilde{S}} T_k$ (implying T_k is not t-complete), then move T_k to immediately precede T_ℓ in $seq(S)$.

By construction, S is equivalent to $\tilde{S}$ and for all $T_k, T_m \in txns(H)$; $T_k \prec_H^{LS} T_m$, $T_k <_S T_m$ We claim that S is a serialization of H. Observe that any two transactions that are complete in H, but not t-complete are not related by real-time order in H. By construction of S, for any transaction $T_k \in txns(H)$, 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 $read_k(X)$ performed by some transaction T_k that returns $v \in V$ in S and let $T_\ell \in txns(H)$ be the earliest transaction in $\tilde{S}$ such that $T_k \prec_H^{LS} T_\ell$. Suppose, by contradiction, that $read_k(X)$ is not legal in S. Thus, there is a committed transaction T_m that performs $write_m(X,v)$ in $\tilde{S}$ such that $T_m = T_\ell$ or $T_\ell <_{\tilde{S}} T_m <_{\tilde{S}} T_k$. Note that, by our assumption, $read_k(X) \prec_H^{RT} tryC_\ell()$. Since $read_k(X)$ must be legal in its local serialization with respect to H and $\tilde{S}$, $read_k(X) \not\prec_H^{RT} tryC_m()$. Thus, $T_m \in Lset_H(T_k)$. Therefore $T_m \neq T_\ell$. Moreover, T_m is complete, and since it commits in $\tilde{S}$, it is also t-complete in H and the last event of T_m precedes the first event of T_ℓ in H, i.e., $T_m \prec_H^{RT} T_\ell$. Hence, T_ℓ cannot precede T_m in $\tilde{S}$—a contradiction.

Observe also that since T_k is complete in H but not t-complete, H does not contain an invocation of $tryC_k()$. Thus, the legality of any other transaction is unaffected by moving T_k to precede T_ℓ 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. □

The proof uses König's Path Lemma [13] formulated as follows. Let G on a rooted directed graph and let v_0 be the root 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 is 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 it has infinitely many vertices.

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 distinct vertices $v_0, v_1, \ldots$, such that v_0 is the* root, *and for every $i \geq 0$, there is an edge from v_i to v_{i+1}.*

Theorem 1. *Under the restriction that in any infinite history H, every transaction $T_k \in txns(H)$ is complete, du-opacity is a limit-closed property.*

Proof. We want to show that the limit H of an infinite sequence of finite ever-extending du-opaque histories is du-opaque. By Corollary 1, we can assume the sequence of du-opaque histories to be $H^0, H^1, \ldots H^i, H^{i+1}, \ldots$ such that for all $i \in \mathbb{N}$, H^{i+1} is the one-event extension of H^i.

We construct a rooted directed graph G_H as follows:

1. The root vertex of G_H is (H^0, S^0) where S^0 and H^0 contain the initial transaction T_0.
2. 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 2: for all $T_k, T_m \in txns(H)$; $T_k \prec_{H^i}^{LS} T_m$ implies $T_k <_{S^i} T_m$. 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.
3. Let $cseq_i(S^j)$, $j \geq i$, denote the subsequence of $seq(S^j)$ restricted to transactions whose last event in H is a response event and it is contained in H^i. 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 $cseq_i(S^i) = cseq_i(S^{i+1})$.

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 $txns(S^{i+1})$, implying that G_H is *finitely branching*.

By Lemma 1, given any serialization S^{i+1} of H^{i+1}, there is 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 2. Since $seq(S^{i+1})$ contains every complete transaction that takes its last step in H in H^i, $cseq_i(S^i) = cseq_i(S^{i+1})$. Therefore, for every vertex (H^{i+1}, S^{i+1}), there is a vertex (H^i, S^i) such that $cseq_i(S^i) = cseq_i(S^{i+1})$. 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 (Lemma 3) to G_H. Since G_H is an infinite connected finitely branching rooted directed graph, we can derive an infinite sequence of distinct vertices

$$\mathscr{L} = (H^0, S^0), (H^1, S^1), \ldots, (H^i, S^i), \ldots$$

such that $cseq_i(S^i) = cseq_i(S^{i+1})$.

The rest of the proof explains how to use $\mathscr{L}$ to construct a serialization of H. We begin with the following claim concerning $\mathscr{L}$.

Claim. For any $j > i$, $cseq_i(S^i) = cseq_i(S^j)$.

Proof. Recall that $cseq_i(S^i)$ is a prefix of $cseq_i(S^{i+1})$, and $cseq_{i+1}(S^{i+1})$ is a prefix of $cseq_{i+1}(S^{i+2})$. Also, $cseq_i(S^{i+1})$ is a subsequence of $cseq_{i+1}(S^{i+1})$. Hence, $cseq_i(S^i)$ is a subsequence of $cseq_{i+1}(S^{i+2})$. But, $cseq_{i+1}(S^{i+2})$ is a subsequence of $cseq_{i+2}(S^{i+2})$. Thus, $cseq_i(S^i)$ is a subsequence of $cseq_{i+2}(S^{i+2})$. Inductively, for any $j > i$, $cseq_i(S^i)$ is a subsequence of $cseq_j(S^j)$. But $cseq_i(S^j)$ is the subsequence of $cseq_j(S^j)$ restricted to complete transactions in H whose last step is in H^i. Thus, $cseq_i(S^i)$ is indeed equal to $cseq_i(S^j)$. □

Let $f : \mathbb{N} \to txns(H)$ be defined as follows: $f(1) = T_0$. For every integer $k > 1$, let

$$i_k = \min\{\ell \in \mathbb{N} | \forall j > \ell : cseq_\ell(S^\ell)[k] = cseq_j(S^j)[k]\}$$

Then, $f(k) = cseq_{i_k}(S^{i_k})[k]$.

Claim. The function f is *total* and *bijective*.

Proof. (Totality and surjectivity)

Since each transaction $T \in txns(H)$ is complete in some prefix H^i of H, for each $k \in \mathbb{N}$, there exists $i \in \mathbb{N}$ such that $cseq_i(S^i)[k] = T$. By Claim 3.4, for any $j > i$, $cseq_i(S^i) = cseq_i(S^j)$. Since a transaction that is complete in H^i w.r.t H is also complete in H^j w.r.t H, it follows that for every $j > i$, $cseq_j(S^j)[k'] = T$, with $k' \geq k$. By construction of G_H and the assumption that each transaction is complete in H, there exists $i \in \mathbb{N}$ such that each $T \in Lset_{H^i}(T)$ is complete in H and its last step is in H^i, and T precedes in S^i every transaction whose first event succeeds the last event of each $T' \in Lset_{H^i}(T)$ in H^i. Indeed, this implies that for each $k \in \mathbb{N}$, there exists $i \in \mathbb{N}$ such that $cseq_i(S^i)[k] = T$; $\forall j > i : cseq_j(S^j)[k] = T$.

This shows that for every $T \in txns(H)$, there are $i, k \in \mathbb{N}$; $cseq_i(S^i)[k] = T$, such that for every $j > i$, $cseq_j(S^j)[k] = T$. Thus, for every $T \in txns(H)$, there is k such that $f(k) = T$.

(Injectivity)

If $f(k)$ and $f(m)$ are transactions at indices k, m of the same $cseq_i(S^i)$, then clearly $f(k) = f(m)$ implies $k = m$. Suppose $f(k)$ is the transaction at index k in some $cseq_i(S^i)$ and $f(m)$ is the transaction at index m in some $cseq_\ell(S^\ell)$. For every $\ell > i$ and $k < m$, if $cseq_i(S^i)[k] = T$, then $cseq_\ell(S^\ell)[m] \neq T$ since $cseq_i(S^i) = cseq_i(S^\ell)$. If $\ell > i$ and $k > m$, it follows from the definition that $f(k) \neq f(m)$. Similar arguments for the case when $\ell < i$ prove that if $f(k) = f(m)$, then $k = m$. □

By Claim 3.4, $\mathscr{F} = f(1), f(2), \ldots, f(i), \ldots$ is an infinite sequence of transactions. Let S be a t-complete t-sequential history such that $seq(S) = \mathscr{F}$ and for each t-complete transaction T_k in H, $S|k = H|k$; and for transaction that is complete, but not t-complete in H, $S|k$ consists of the sequence of events in $H|k$, immediately followed by $tryA_k() \cdot A_k$. Clearly, there is a completion of H that is equivalent to S.

Let $\mathscr{F}^i$ be the prefix of $\mathscr{F}$ of length i, and $\widehat{S^i}$ be the prefix of S such that $seq(\widehat{S^i}) = \mathscr{F}^i$.

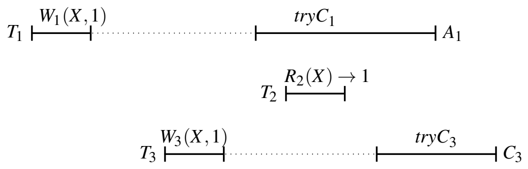

Fig. 3. A history that is opaque, but not du-opaque

Claim. Let $\widehat{H}_i^j$ be a subsequence of H^j reduced to transactions $T_k \in txns(\widehat{S}^i)$ such that the last event of T_k in H is a response event and it is contained in H^j. Then, for every i, there is j such that $\widehat{S}^i$ is a serialization of $\widehat{H}_i^j$.

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

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 $tryC_k \cdot A_k$ after its last event. By Claim 3.4, 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. □

Theorem 1 implies the following:

Corollary 2. *Let M be a TM implementation that ensures that in every infinite history H of M, every transaction $T \in txns(H)$ is complete in H. Then, M is du-opaque if and only if every finite history of M is du-opaque.*

4 Du-Opacity vs. Opacity

We now compare our deferred-update requirement with the conventional TM correctness property of opacity [7].

Theorem 2. *Du-opacity $\subsetneq$ Opacity.*

Proof. We first claim that every finite du-opaque history is opaque. Let H be a finite du-opaque history. By definition, there is 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 4, every infinite du-opaque history is also opaque.

To show that the inclusion is strict, we present an an opaque history that is not du-opaque. Consider the finite history H depicted in Figure 3: transaction T_2 performs a $read_2(X)$ that returns the value 1. Observe that $read_2(X) \rightarrow 1$ is concurrent to $tryC_1$, but precedes $tryC_3$ in real-time order. Although $tryC_1$ returns A_1 in H, the response of $read_2(X)$ can be justified since T_3 concurrently writes 1 to X and commits. Thus, $read_2(X) \rightarrow 1$ *reads-from* transaction T_2 in any serialization of H, but since $read_2(X) \prec_H^{RT} tryC_3$, H is not du-opaque even though each of its prefixes is final-state opaque.

We now formally prove that H is opaque. We proceed by examining every prefix of H.

1. Each prefix up to the invocation of $read_2(X)$ is trivially final-state opaque.
2. Consider the prefix, H^i of H where the i^{th} event is the response of $read_2(X)$. 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 $tryC_1()$. 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 required final-state serialization S of H is specified by $seq(S) = T_1, T_3, T_2$ in which T_1 is aborted while T_3 is committed in S (the position of T_1 in the serialization does not affect legality). Consider $read_2(X)$ in S; since $H^{2,X}$, the prefix of H up to the response of $read_2(X)$ does not contain an invocation of $tryC_3()$, the local serialization of $read_2(X)$ with respect to H and S, $S_H^{2,X}$ is $T_1 \cdot read_2(X)$. But $read_2(X)$ is not legal in $S_H^{2,X}$, which is a contradiction. Thus, H is not du-opaque. $\square$

4.1 The Unique-Write Case

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 Opacity$_{uw}$ $\subseteq$ Opacity, be a property defined as follows:

1. an infinite opaque history $H \in$ Opacity$_{uw}$ if and only if every transaction $T \in txns(H)$ is complete in H, and
2. an opaque history $H \in$ Opacity$_{uw}$ if and only if for every pair of write operations $write_k(X, v)$ and $write_m(X, v')$, $v \neq v'$.

Theorem 3. *Opacity$_{uw}$ =du-opacity.*

Proof. We show first that every finite history $H \in$Opacity$_{uw}$ is also du-opaque. Let H be any finite opaque history such that for every pair of write operations $write_k(X, v)$ and $write_m(X, v)$, performed by transactions $T_k, T_m \in txns(H)$, respectively, $v \neq v'$.

Since H is opaque, there is a final-state serialization S of H. Suppose by contradiction that H is not du-opaque. Thus, there is a $read_k(X)$ that returns a value $v \in V$ in S that is not legal in $S_H^{k,X}$, the local serialization of $read_k(X)$ with respect to H and S. Let $H^{k,X}$ and $S^{k,X}$ denote the prefixes of H and S, respectively, up to the response of $read_k(X)$ in H and S. Recall that $S_H^{k,X}$, the local serialization of $read_k(X)$ with respect to H and S, is the subsequence of $S^{k,X}$ that does not contain events of any transaction $T_i \in txns(H)$ so that the invocation of $tryC_i()$ is not in $H^{k,X}$. Since $read_k(X)$ is legal in S, there is a

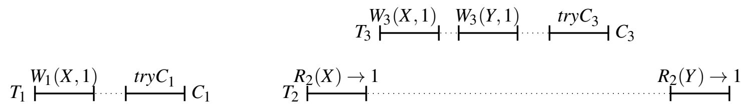

Fig. 4. A sequential du-opaque history, which is not opaque by the definition of [6]

committed transaction $T_m \in txns(H)$ 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 $S_H^{k,X}$, the only possibility is that $read_k(X) \prec_H^{RT} tryC_m()$. Under the assumption of unique writes, there does not exist any other transaction $T_j \in txns(H)$ that performs $write_j(X,v)$. Consequently, there does not exist any $\overline{H}^{k,X}$ (some completion of $H^{k,X}$) and a t-complete t-sequential history S', such that S' is equivalent to $\overline{H}^{k,X}$ and S' contains any committed transaction that writes v to X. This is, $H^{k,X}$ is not final-state opaque. However, since H is opaque, every prefix of H must be final-state opaque, which is a contradiction.

By Definition 4, an infinite history H is opaque if every finite prefix of H is final-state opaque. Theorem 1 now implies that Opacity$_{uw}$ $\subseteq$ du-Opacity.

Definition 4 and Corollary 1 imply that du-Opacity $\subseteq$ Opacity$_{uw}$. □

4.2 The Sequential-History Case

The deferred-update semantics was mentioned by Guerraoui et al. [6] and later adopted by Kuznetsov and Ravi [14]. In both papers, opacity is only defined for 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*: in these definitions, a history H is opaque if there is 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. As we observed in Figure 4, this definition is not equivalent to opacity even for sequential histories.

The property considered in [6, 14] is strictly stronger than du-opacity: the sequential history H in Figure 4 is du-opaque (and consequently opaque by Theorem 2): a du-opaque serialization (in fact the only possible one) for this history is T_1, T_3, T_2. However, in the restriction of opacity defined above, T_2 must precede T_3 in any serialization, since the response of $read_2(X)$ precedes the invocation of tryC$_3$().

5 Du-Opacity vs. Other Deferred-Update Criteria

In this section, we compare du-opacity to other TM correctness conditions, restricted to provide the deferred-update semantics. We first discuss the stronger TMS2 property [5], and then describe deferred-update versions of conditions weaker than opacity, VWC [12] and TMS1 [5].

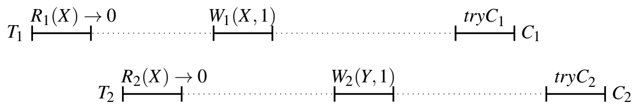

Fig. 5. A history that is du-opaque, but not TMS2 [5]

5.1 TMS2

Transactional Memory Specification (TMS) 1 and 2 were formulated in I/O automata [5]. Following [2], we adapt these definitions to our framework and explicitly introduce the deferred-update requirement. We start with TMS2, a restriction of opacity, and discuss TMS1, a relaxation of du-opacity, in Section 5.3.

Definition 7 (du-TMS2). *A history H is* du-TMS2 *if there is a legal t-complete t-sequential history S equivalent to some completion, $\overline{H}$ of H such that*

1. *for any two transactions $T_k, T_m \in txns(H)$, such that T_m is a committed updating transaction, if $C_k \prec_H^{RT} tryC_m$ or $A_k \prec_H^{RT} tryC_m$, then $T_k \prec_S T_m$, and*
2. *for any two transactions $T_k, T_m \in txns(H)$, if $T_k \prec_H^{RT} T_m$, then $T_k <_S T_m$, and*
3. *each $read_k(X)$ in S that does not return A_k is legal in $S_H^{k,X}$.*

We refer to S as the du-TMS2 serialization of H.

It has been shown [15] that TMS2 is a strictly stronger property than Opacity, i.e., TMS2 $\subsetneq$ Opacity. We now show that du-TMS2 is strictly stronger than du-opacity. Indeed, from Definition 7, we observe that every history that is du-TMS2 is also du-opaque. The following proposition completes the proof.

Proposition 2. *There is a history that is du-opaque, but not du-TMS2.*

Proof. Figure 5 depicts a history H that is du-opaque, but not du-TMS2. Indeed, there is a du-opaque serialization S of H such that $seq(S) = T_2, T_1$. On the other hand, since T_1 commits before T_2, T_1 must precede T_2 in any du-TMS2 serialization, there does not exist any such serialization that ensures every t-read is legal. Thus, H is not du-TMS2. $\square$

Theorem 4. *Du-TMS2 is prefix-closed.*

Proof. Let H be any du-TMS2 history. Then, H is also du-opaque. By Corollary 1, for every $i \in \mathbb{N}$, there is a du-opaque serialization S^i for H^i. We now need to prove that, for any two transactions $T_k, T_m \in txns(H^i)$, such that T_m is a committed updating transaction, if $C_k \prec_{H^i}^{RT} tryC_m$ or $A_k \prec_{H^i}^{RT} tryC_m$, there is a du-opaque serialization S^i with the restriction that $T_k \prec_{S^i} T_m$.

Suppose by contradiction that there exist transactions $T_k, T_m \in txns(H^i)$, such that T_m is a committed updating transaction and $C_k \prec_{H^i}^{RT} tryC_m$ or $A_k \prec_{H^i}^{RT} tryC_m$, but T_m must precede T_k in any du-opaque serialization S^i. Since $T_m \not\prec_{H^i}^{RT} T_k$, the only possibility is that T_m performs $write_m(X, v)$ and there is $read_k(X) \rightarrow v$. However, by our assumption,

$write_k(X,v) \prec^{RT}_{H^i} tryC_m$: thus, $read_k(X)$ is not legal in its local serialization with respect to H^i and S^i—contradicting the assumption that S^i is a du-opaque serialization of H^i. Thus, there is a du-TMS2 serialization for H^i, proving that du-TMS2 is a prefix-closed property. □

Proposition 3. *Du-TMS2 is not limit-closed.*

Proof. The counter-example to establish that du-opacity is not limit-closed (Figure 2) also shows that du-TMS2 is not limit-closed: all histories discussed in the counter-example are in du-TMS2. □

5.2 Virtual World Consistency (VWC)

Intuitively, VWC [12] and TMS1 [5] achieve intuitively understood safety of each transaction or response, without enforcing a single serialization. Both definitions use the following "deferred-update" version of *strict serializability* [18]:

Definition 8 (Strict serializability). *A finite history H is* strictly serializable *if there is a legal t-complete t-sequential history S, such that*

1. *there is a completion $\overline{H}$ of H, such that S is equivalent to $cseq(\overline{H})$, where $cseq(\overline{H})$ is the subsequence of $\overline{H}$ reduced to committed transactions in $\overline{H}$,*
2. *for any two transactions $T_k, T_m \in txns(H)$, if $T_k \prec^{RT}_H T_m$, then T_k precedes T_m in S, and*
3. *each $read_k(X)$ in S that does not return A_k is legal in $S^{k,X}_H$.*

We refer to S as the (strictly serializable) serialization of H.

Notice that every du-opaque history is strictly serializable, but not vice-versa. The following result will be instrumental for understanding the properties of du-VWC and du-TMS1.

Theorem 5. *Strict serializability is a safety property.*

Proof. (Sketch) Observe that any serialization of a finite history H does not include events of any transaction that has not invoked $tryC$ in H.

To show prefix-closure, a proof almost identical to that of Lemma 1 implies that, given a strictly serializable history H and a serialization S, there is a serialization S' of H' (H' is some prefix of H) such that $seq(S')$ is a prefix of $seq(S)$.

Consider an infinite sequence of finite histories

$$H^0, \ldots, H^i, H^{i+1}, \ldots,$$

where H^{i+1} is a one-event extension of H^i, we prove that the infinite limit H of this ever-extending sequence is strictly serializable. As in Theorem 1, we construct an infinite rooted directed graph G_H: a vertex is a tuple (H^i, S^i) (note that for each $i \in \mathbb{N}$, there are several such vertices of this form), where S^i is a serialization of H^i and there is an edge from (H^i, S^i) to (H^{i+1}, S^{i+1}) if $seq(S^i)$ is a prefix of $seq(S^{i+1})$. The resulting graph is finitely branching since the out-degree of a vertex is bounded by the number of possible serializations of a history. Observe that for every vertex (H^{i+1}, S^{i+1}), there is a vertex $H^i, S^i)$ such that $seq(S^i)$ is a prefix of $seq(S^{i+1})$. Thus, G_H is connected since we can iteratively construct a path from the root (H^0, S^0) to every vertex (H^i, S^i) in G_H. Applying König's Path Lemma to G_H, we obtain an infinite sequence of distinct vertices, $(H^0, S^0), (H^1, S^1), \ldots, (H^i, S^i), \ldots$. Then, $S = \lim_{i \to \infty} S_i$ gives the desired serialization of H. □

Virtual World Consistency (VWC) [12] was proposed as a relaxation of opacity (in our case, du-opacity), where each aborted transaction should be consistent with its *causal past* (but not necessarily with a serialization formed by committed transactions). Intuitively, a transaction T_1 causally precedes T_2 if T_2 reads a value written and committed by T_1. The original definition [12] required that no two write operations are ever invoked with the same argument (the *unique-writes* assumption). Therefore, the causal precedence is unambiguously identified for each transactional read. Below we give a more general definition.

Given a t-sequential legal history S and transactions $T_i, T_j \in txns(S)$, we say that T_i *reads* X *from* T_j if (1) T_i reads v in X and (2) T_j is the last committed transaction that writes v to X and precedes T_i in S.

Now consider a (not necessarily t-sequential) history H. We say that T_i *could have read* X *from* T_j in H if T_j writes a value v to a t-object X, T_i reads v in X, and $read_i(X) \not\prec_H^{RT} tryC_j()$.

Given $\mathscr{T} \subseteq txns(H)$, let $H^{\mathscr{T}}$ denote the subsequence of H restricted to events of transactions in $\mathscr{T}$.

Definition 9 (du-VWC). *A finite history H is* du-virtual-world consistent *if it is strictly serializable, and for every aborted or t-incomplete transaction $T_i \in txns(H)$, there is $\mathscr{T} \subseteq txns(H)$ including T_i and a t-sequential t-complete legal history S such that:*

1. *S is equivalent to a completion of $H^{\mathscr{T}}$,*
2. *For all $T_j, T_k \in txns(S)$, if T_j reads X from T_k in S, then T_j could have read X from T_k in H,*
3. *S respects the per-process order of H: if T_j and T_k are executed by the same process and $T_j \prec_H^{RT} T_k$, then $T_j \prec_S T_k$.*

We refer to S as a du-VWC serialization for T_i in H.

Intuitively, with every t-read on X performed by T_i in H, the du-VWC serialization S associates some transaction T_j from which T_i could have read the value of X. Recursively, with every read performed by T_j, S associates some T_m from which T_j could have read, etc. Altogether, we get a "plausible" causal past of T_i that constitutes a serial history. Notice that to ensure deferred-update semantics, we only allow a transaction T_j to read from a transaction T_k that invoked $tryC_k$ by the time of the read operation of T_j.

We now prove that du-VWC is a strictly weaker property than du-opacity. Since du-TMS2 is strictly weaker than du-opacity (cf. Section 5.1), it follows that Du-TMS2 $\subsetneq$ du-VWC.

Theorem 6. *Du-opacity $\subsetneq$ du-VWC.*

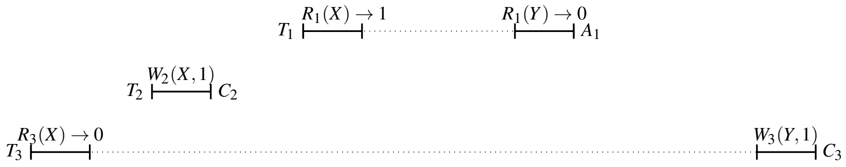

Fig. 6. A history that is du-VWC, but not du-opaque

Proof. If a history H is du-opaque, then there is a du-opaque serialization S equivalent to $\overline{H}$, where $\overline{H}$ is some completion of H. By construction, S is a total-order on the set of all transactions that participate in S. Trivially, by taking $\mathscr{T} = txns(H)$, we derive that S is a du-VWC serialization for every aborted or t-incomplete transaction $T_i \in txns(H)$. Indeed, S respects the real-time order and, thus, the per-process order of H. Since S respects the deferred-update order in H, every t-read in S "could have happened" in H.

To show that the inclusion is strict, Figure 6 depicts a history H that is du-VWC, but not du-opaque. Clearly, H is strictly serializable. Here T_2, T_1 is the required du-VWC serialization for aborted transaction T_1. However, H has no du-opaque serialization. □

Theorem 7. *Du-VWC is a safety property.*

Proof. By Definition 9, a history H is du-VWC if and only if H is strictly serializable and there is a du-VWC serialization for every transaction $T_i \in txns(H)$ that is aborted or t-incomplete in H.

To prove prefix-closure, recall that strict serializability is a prefix-closed property (Theorem 5). Therefore, any du-VWC serialization S for a transaction T_i in history H is also a du-VWC serialization S for a transaction T_i in any prefix of H that contains events of T_i.

To prove limit-closure, consider an infinite sequence of du-VWC histories H^0, H^1, $\ldots$, H^i, H^{i+1} , $\ldots$, where each H^{i+1} is the one-event extension of H^i and prove that the infinite limit, H of this sequence is also a du-VWC history. Theorem 5 establishes that there is a strictly serializable serialization for H.

Since, for all $i \in \mathbb{N}$, H^i is du-VWC, for every transaction T_i that is t-incomplete or aborted in H^i, there is a VWC serialization for T_i. Consequently, there is a du-VWC serialization for every aborted or incomplete transaction T_i in H. □

5.3 TMS1

Given a history H, TMS1 requires us to justify the behavior of all committed transactions in H by a legal t-complete t-sequential history that preserves the real-time order in H (strict serializability), and to justify the response of each complete t-operation performed in H by a legal t-complete t-sequential history S. The t-sequential history S used to justify a complete t-operation $op_{i,k}$ (the i^{th} t-operation performed by transaction T_k) includes T_k and a subset of transactions from H whose operations justify $op_{i,k}$. (Our description follows [2].)

Let $H^{k,i}$ denote the prefix of a history H up to (and including) the response of i^{th} t-operation $op_{k,i}$ of transaction T_k. We say that a history H'' is a *possible past* of $H^{k,i}$ if

H'' is a subsequence of $H^{k,i}$ and consists of all events of transaction T_k and all events from some subset of committed transactions and transactions that have invoked *tryC* in $H^{k,i}$ such that if a transaction $T \in H''$, then for a transaction $T' \prec^{RT}_{H^{k,i}} T$, $T' \in H''$ if and only if T' is committed in $H^{k,i}$. Let $cTMSpast(H, op_{k,i})$ denote the set of possible pasts of $H^{k,i}$.

For any history $H'' \in cTMSpast(H, op_{k,i})$, let $ccomp(H'')$ denote the history generated from H'' by the following procedure: for all $m \neq k$, replace every event A_m by C_m and complete every incomplete $tryC_m$ with including C_m at the end of H''; include $tryC_k \cdot A_k$ at the end of H''.

Definition 10 (du-TMS1). *A history H satisfies* du-TMS1 *if*

1. *H is strictly serializable, and*
2. *for each complete t-read $op_{i,k}$ that returns a non-A_k response in H, there exist a legal t-complete t-sequential history S and a history H′ such that:*
   – $H' = ccomp(H'')$, *where* $H'' \in cTMSpast(H, op_{k,i})$
   – *H' is equivalent to S*
   – *for any two transactions T_k and T_m in H', if $T_k \prec^{RT}_{H'} T_m$ then $T_k <_S T_m$*

We refer to S as the du-TMS1 serialization for $op_{i,k}$.

Theorem 8. *Du-TMS1 is a safety property.*

Proof. A history H is du-TMS1 if and only if H is strictly serializable and there is a du-TMS1 serialization for every t-operation $op_{k,i}$ that does not return A_k in H.

To see that du-TMS1 is prefix closed, recall that strict serializability is a prefix-closed property. Let H be any du-TMS1 history and H^i, any prefix of H. We now need to prove that, for every t-operation $op_{k,i} \neq tryC_k$ that returns a non-A_k response in H^i, there is a du-TMS1 serialization for $op_{k,i}$. But this is immediate since the du-TMS1 serialization for $op_{i,k}$ in H is also the required du-TMS1 serialization for $op_{k,i}$ in H^i.

To see that du-TMS1 is limit closed, consider an infinite sequence

$$H^0, H^1, \ldots H^i, H^{i+1}, \ldots$$

of finite du-TMS1 histories, such that H^{i+1} is a one-event extension of H^i. Let let H be the corresponding infinite limit history. We want to show that H is also du-TMS1.

Since strict serializability is a limit-closed property (Theorem 5), H is strictly serializable. By assumption, for all $i \in \mathbb{N}$, H^i is du-TMS1. Thus, for every transaction T_i that participates in H^i, there is a du-TMS1 serialization $S^{i,k}$ for each t-operation $op_{k,i}$. But $S^{i,k}$ is also the required du-TMS1 serialization for $op_{k,1}$ in H. The claim follows. □

It has been shown [15] that Opacity is a strictly stronger property than du-TMS1, that is, Opacity $\subsetneq$ du-TMS1. Since Du-Opacity $\subsetneq$ Opacity (Theorem 2) it follows that Du-Opacity $\subsetneq$ du-TMS1. On the other hand, du-TMS1 is incomparable to du-VWC, as demonstrated by the following examples.

Proposition 4. *There is a history that is du-TMS1, but not du-VWC.*

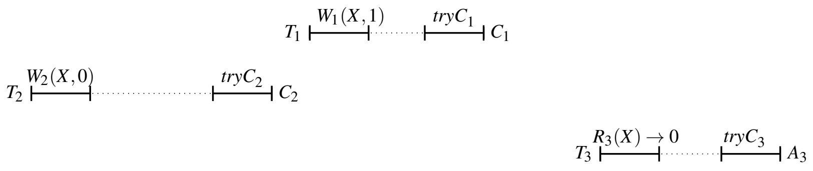

Fig. 7. A history which is du-VWC but not du-TMS1

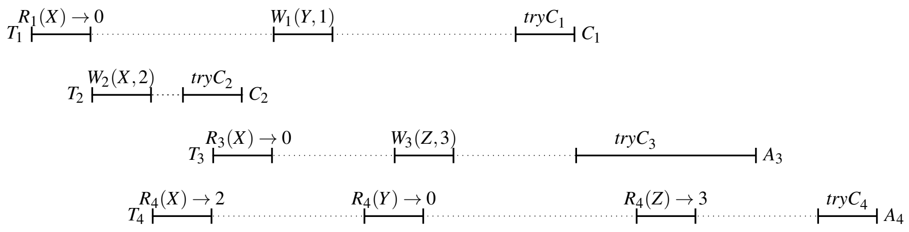

Fig. 8. A history which is du-TMS1 but not du-VWC

Proof. Figure 8 depicts a history H that is du-TMS1, but not du-VWC. Observe that H is strictly serializable. To prove that H is du-TMS1, we need to prove that there is a TMS1 serialization for each t-read that returns a non-abort response in H. Clearly, the serialization in which only T_3 participates is the required TMS1 serialization for $read_3(X) \to 0$. Now consider the aborted transaction T_4. The TMS1 serialization for $read_4(X) \to 2$ is T_2, T_4, while the TMS1 serialization that justifies the response of $read_4(Y) -> 0$ includes just T_4 itself. The only nontrivial t-read whose response needs to be justified is $read_4(Z) \to 3$. Indeed, $tryC_3$ overlaps with $read_4(Z)$ and thus, the response of $read_4(Z)$ can be justified by choosing transactions in $cTMSpart(H, read_4(Z))$ to be $\{T_3, T_2, T_4\}$ and then deriving a TMS1 serialization $S = T_3, T_2, T_4$ for $read_4(Z) \to 3$ in which $tryC_3$ may be completed by including the commit response.

However, H is not du-VWC. Consider transaction T_3 which returns A_3 in H: T_3 must be aborted in any serialization equivalent to some direct causal past of T_4. But $read_4(Z)$ returns the value 3 that is written by T_3. Thus, $read_4(Z)$ cannot be legal in any du-VWC serialization for T_4. □

Proposition 5. *There is a history that is du-VWC, but not du-TMS1.*

Proof. Figure 7 depicts a history H that is du-VWC, but not du-TMS1. Clearly, H is strictly serializable. Observe that T_3 could have read only from T_1 in H (T_1 writes the value 0 to X that is returned by $read_3(X)$). Therefore, T_1, T_3 is the required du-VWC serialization for aborted transaction T_3.

However, H is not du-TMS1: since both transactions T_1 and T_2 are committed and precede T_3 in real-time order, they must be included in any du-TMS1 serialization for $read_3(X) \to 0$. But there is no such du-TMS1 serialization that would ensure the legality of $read_3(X)$. □

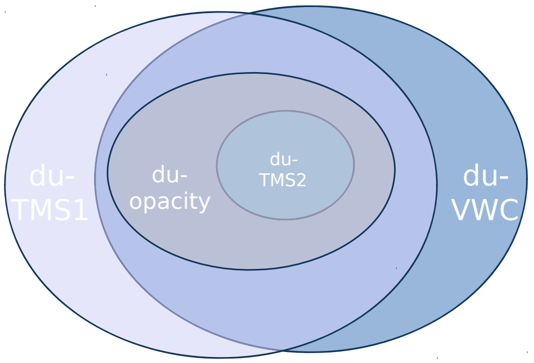

Fig. 9. Relations between TM consistency definitions

6 Concluding Remarks

The properties discussed in this paper explicitly preclude reading from a transaction that has not yet invoked *tryC*, which makes them prefix-closed and facilitates their verification. We believe that this constructive definition is useful to TM practitioners, since it streamlines possible implementations of t-read and tryC operations.

We showed that du-opacity is limit-closed under the restriction that every operation eventually terminates, while du-VWC and du-TMS1 are (unconditionally) limit-closed, which makes them safety properties [16].

Figure 9 summarizes the containment relations between the properties discussed in this chapter: opacity, du-opacity, du-VWC, du-TMS1 and du-TMS2.

Linearizability [4, 11], when applied to objects with *finite nondeterminism* (i.e., an operation applied to a given state may produce only finitely many outcomes) sequential specifications is a safety property [8, 16]. Recently, it has been shown [8] that linearizability is not limit-closed if the implemented object may expose infinite nondeterminism [8], that is, an operation applied to a given state may produce infinitely many different outcomes. The limit-closure proof (cf. Theorem 1), using König's lemma, cannot be applied with infinite non-determinism, because the out-degree of the graph G_H, constructed for the limit infinite history H, is not finite.

In contrast, the TM abstraction is *deterministic*, since reads and writes behave deterministically in serial executions, yet du-opacity is not limit-closed. It turns out that the graph G_H for the counter-example history H in Figure 2 is not connected. For example, one of the finite prefixes of H can be serialized as T_3, T_1, T_2, but no prefix has a serialization T_3, T_1 and, thus, the root is not connected to the corresponding vertex of G_H. Thus, the precondition of König's lemma does not hold for G_H: the graph is in fact an infinite set of isolated vertices. This is because du-opacity requires even incomplete reading transactions, such as T_2, to appear in the serialization, which is not the case for linearizability, where incomplete operations may be removed from the linearization.

References

1. Alpern, B., Schneider, F.B.: Defining liveness. Information Processing Letters 21(4), 181–185 (1985)
2. Attiya, H., Gotsman, A., Hans, S., Rinetzky, N.: Safety of live transactions in transactional memory: TMS is necessary and sufficient. In: Kuhn, F. (ed.) DISC 2014. LNCS, vol. 8784, pp. 376–390. Springer, Heidelberg (2014)
3. Attiya, H., Hans, S., Kuznetsov, P., Ravi, S.: Safety of deferred update in transactional memory. In: ICDCS, pp. 601–610 (2013)
4. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, 2nd edn. Wiley Interscience (2004)
5. Doherty, S., Groves, L., Luchangco, V., Moir, M.: Towards formally specifying and verifying transactional memory. Formal Asp. Comput. 25(5), 769–799 (2013)
6. Guerraoui, R., Henzinger, T.A., Singh, V.: Permissiveness in transactional memories. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 305–319. Springer, Heidelberg (2008)
7. Guerraoui, R., Kapalka, M.: Principles of Transactional Memory, Synthesis Lectures on Distributed Computing Theory. Morgan and Claypool (2010)
8. Guerraoui, R., Ruppert, E.: Linearizability is not always a safety property. In: Noubir, G., Raynal, M. (eds.) NETYS 2014. LNCS, vol. 8539, pp. 57–69. Springer, Heidelberg (2014)
9. Hadzilacos, V.: A theory of reliability in database systems. J. ACM 35(1), 121–145 (1988)
10. Herlihy, M., Moss, J.E.B.: Transactional memory: Architectural support for lock-free data structures. SIGARCH Comput. Archit. News 21(2), 289–300 (1993)
11. Herlihy, M., Wing, J.M.: Linearizability: A correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst. 12(3), 463–492 (1990)
12. Imbs, D., Raynal, M.: Virtual world consistency: A condition for STM systems (with a versatile protocol with invisible read operations). Theor. Comput. Sci. 444 (July 2012)
13. König, D.: Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe. Akad. Verlag (1936)
14. Kuznetsov, P., Ravi, S.: On the cost of concurrency in transactional memory. CoRR, abs/1103.1302 (2011)
15. Lesani, M., Luchangco, V., Moir, M.: Putting opacity in its place. In: WTTM (2012)
16. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann (1996)
17. Owicki, S.S., Lamport, L.: Proving liveness properties of concurrent programs. ACM Trans. Program. Lang. Syst. 4(3), 455–495 (1982)
18. Papadimitriou, C.H.: The serializability of concurrent database updates. J. ACM 26, 631–653 (1979)
19. Shavit, N., Touitou, D.: Software transactional memory. In: PODC 1995, pp. 204–213 (1995)