On the Semantics of Snapshot Isolation

Azalea Raad¹, Ori Lahav², and Viktor Vafeiadis¹

¹ MPI-SWS, Germany
² Tel Aviv University, Israel
{azalea,viktor}@mpi-sws.org
orilahav@tau.ac.il

Abstract. Snapshot isolation (SI) is a standard transactional consistency model used in databases, distributed systems and software transactional memory (STM). Its semantics is formally defined both declaratively as an acyclicity axiom, and operationally as a concurrent algorithm with memory bearing timestamps.

We develop two simpler equivalent operational definitions of SI as lock-based reference implementations that do not use timestamps. Our first locking implementation is prescient in that requires a priori knowledge of the data accessed by a transaction and carries out transactional writes eagerly (in-place). Our second implementation is non-prescient and performs transactional writes lazily by recording them in a local log and propagating them to memory at commit time. Whilst our first implementation is simpler and may be better suited for developing a program logic for SI transactions, our second implementation is more practical due to its non-prescience. We show that both implementations are sound and complete against the declarative SI specification and thus yield equivalent operational definitions for SI.

We further consider, for the first time formally, the use of SI in a context with racy non-transactional accesses, as can arise in STM implementations of SI. We introduce robust snapshot isolation (RSI), an adaptation of SI with similar semantics and guarantees in this mixed setting. We present a declarative specification of RSI as an acyclicity axiom and analogously develop two operational models as lock-based reference implementations (one eager, one lazy). We show that these operational models are both sound and complete against the declarative RSI model.

1 Introduction

Transactions are the de facto synchronisation mechanism in databases and geo-replicated distributed systems, and are thus gaining adoption in the shared-memory setting via software transactional memory (STM) [33,20]. In contrast to other synchronisation mechanisms, transactions readily provide atomicity, isolation, and consistency guarantees for sequences of operations, allowing programmers to focus on the high-level design of their systems.

However, providing these guarantees comes at a significant cost. As such, various transactional consistency models in the literature trade off consistency
guarantees for better performance. At nearly the one end of the spectrum, we have serialisability [28], which requires transactions to appear to have been executed in some total order. Serialisability provides strong guarantees, but is widely considered too expensive to implement. The main problem is that two conflicting transactions (e.g. one reading from and one updating the same datum) cannot both execute and commit in parallel.

Consequently, most major databases, both centralised (e.g. Oracle and MS SQL Server) and distributed [15,32,29], have opted for a slightly weaker model called snapshot isolation (SI) [7] as their default consistency model. SI has much better performance than serialisability by allowing conflicting transactions to execute concurrently and commit successfully as long as they do not have a write-write conflict. This in effect allows reads of SI transactions to read from an earlier memory snapshot than the one affected by their writes, and permits the write skew anomaly [11] depicted in Fig. 1. Besides this anomaly, however, SI is essentially the same as serialisability: Cerone et al. [11] provide a widely applicable condition under which SI and serialisability coincide for a given set of transactions. For these reasons, SI has also started gaining adoption in the generic programming language setting via STM implementations [1,16,8,26,25] that provide SI semantics for their transactions.

The formal study of SI, however, has so far not accounted for the more general STM setting in which both transactions and uninstrumented non-transactional code can access the same memory locations. While there exist two equivalent definitions of SI—one declarative in terms of an acyclicity constraint [10,11] and one operational in terms of an optimistic multi-version concurrency control algorithm [7]—neither definition supports mixed-mode (i.e. both transactional and non-transactional) accesses to the same locations. Extending the definitions to do so is difficult for two reasons: (1) the operational definition attaches a timestamp to every memory location, which heavily relies on the absence of non-transactional accesses; and (2) there are subtle interactions between the transactional implementation and the weak memory model underlying the non-transactional accesses.

In this article, we address these limitations of SI. We develop two simple lock-based reference implementations for SI that do not use timestamps. Our first implementation is prescient [19] in that it requires a priori knowledge of the data accessed by a transaction, and performs transactional writes eagerly (in-place). Our second implementation is non-prescient and carries out transactional writes lazily by first recording them in a local log and subsequently propagating them to memory at commit time. Our first implementation is simpler and may be better suited for understanding and developing a program logic for SI transactions, whilst our second implementation is more practical due to its non-prescience. We show that both implementations are sound and complete against the declarative SI specification and thus yield equivalent operational definitions for SI.

We then extend both our eager and lazy implementations to make them robust under uninstrumented non-transactional accesses, and characterise declaratively
the semantics we obtain. We call this extended model robust snapshot isolation (RSI) and show that it gives reasonable semantics with mixed-mode accesses.

To provide SI semantics, instead of timestamps, our implementations use multiple-readers-single-writer (MRSW) locks. They acquire locks in reader mode to take a snapshot of the memory locations accessed by a transaction and then promote the relevant locks to writer mode to enforce an ordering on transactions with write-write conflicts. As we discuss in §4, the equivalence of the RSI implementation and its declarative characterisation depends heavily upon the axiomatisation of MRSW locks: here, we opted for the weakest possible axiomatisation that does not order any concurrent reader lock operations and present an MRSW lock implementation that achieves this.

Outline In §2 we present an overview of our contributions by describing our reference implementations for both SI and RSI. In §3 we define the declarative framework for specifying STM programs. In §4 we present the declarative SI specification against which we demonstrate the soundness and completeness of our SI implementations. In §5 we formulate a declarative specification for RSI and demonstrate the soundness and completeness of our RSI implementations. We discuss related and future work in §6.3

2 Background and Main Ideas

As noted earlier, the key challenge in specifying STM transactions lies in accounting for the interactions between mixed-mode accesses to the same data. One simple approach is to treat each non-transactional access as a singleton mini-transaction and to provide strong isolation [27,9], i.e. full isolation between transactional and non-transactional code. This, however, requires instrumenting non-transactional accesses to adhere to same access policies as transactional ones (e.g. acquiring the necessary locks), which incurs a substantial performance penalty for non-transactional code. A more practical approach is to enforce isolation only amongst transactional accesses, an approach known as weak isolation [27,9], adopted by the relaxed transactions of C++ [2].

As our focus is on STMs with SI guarantees, instrumenting non-transactional accesses is not feasible. In particular, as we expect many more non-transactional accesses than transactional ones, we do not want to incur any performance degradation on non-transactional code when executed in parallel with transactional code. As such, we opt for an STM with SI guarantees under weak isolation. Under weak isolation, however, transactions with explicit abort instructions are problematic as their intermediate state may be observed by non-transactional code. As such, weakly isolated STMs (e.g. C++ relaxed transactions [2]) often forbid explicit aborts altogether. Throughout our development we thus make two simplifying assumptions: (1) transactions are not nested; and (2) there are no explicit abort instructions, following the example of weakly isolated relaxed

3 A full version of this article is available at [31].
### Snapshot Isolation (SI)

The initial model of SI in [7] is described informally in terms of a multi-version concurrent algorithm as follows. A transaction \( T \) proceeds by taking a snapshot \( S \) of the shared objects. The execution of \( T \) is then carried out locally: read operations query \( S \) and write operations update \( S \). Once \( T \) completes its execution, it attempts to commit its changes and succeeds only if it is not write-conflicted. Transaction \( T \) is write-conflicted if another committed transaction \( T' \) has written to a location also written to by \( T \) and the commit-timestamp of \( T' \) is in the execution interval of \( T \) \([t_0, t_c]\). To realise this, the shared state is represented as a series of multi-versioned objects: each object is associated with a history of several versions at different timestamps. In order to obtain a snapshot, a transaction \( T \) chooses a start-timestamp \( t_0 \), and reads data from the committed state as of \( t_0 \), ignoring updates after \( t_0 \). That is, updates committed after \( t_0 \) are invisible to \( T \) and its changes become visible to all other transactions that take a snapshot thereafter.

Transactions of C++. As we describe later in §2.3, it is straightforward to lift the latter restriction (2) for our lazy implementations.

For non-transactional accesses, we naturally have to pick some consistency model. For simplicity and uniformity, we pick the release/acquire (RA) subset of the C++ memory model [6,23], a well-behaved platform-independent memory model, whose compilation to x86 requires no memory fences.

### Figures

<table>
<thead>
<tr>
<th>T1:</th>
<th>T2:</th>
<th>T1:</th>
<th>T2:</th>
<th>T1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>([a := x; #0] ) ( x := a + 1; )</td>
<td>([b := x; #0] ) ( x := b + 1; )</td>
<td>([a := x; #0] ) ( y := 1; )</td>
<td>([b := y; #0] ) ( x := 1; )</td>
<td>([y := 1; ) ( T2: ([b := y; #0] ) ( x := 1; )</td>
</tr>
<tr>
<td>( \text{(LU)} \text{ Lost Update} )</td>
<td>( \text{SI: } \text{X} )</td>
<td>( \text{SI: } \text{✓} )</td>
<td>( \text{SI: } \text{✓} )</td>
<td>( \text{SI: } \text{X} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T1:</th>
<th>T2:</th>
<th>T1:</th>
<th>T2:</th>
<th>T3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := 1; ) ( y := 1; ) ( a := y; #2 ) ( y := 2; ) ( x := 1; )</td>
<td>([b := x; #0 ) ( y := 1; ) ( b := z; ) ( y := 1; ) ( b := y; #0 ) ( d := x; #0 ) ( x := 1; )</td>
<td>( \text{(SBT) Store Buffering} ) ( \text{RSI: } \text{✓} )</td>
<td>( \text{(MPT) Message Passing} ) ( \text{RSI: } \text{X} )</td>
<td></td>
</tr>
<tr>
<td>( \text{(LU2) Lost Update Variant} )</td>
<td>( \text{SI: } \text{X} )</td>
<td>( \text{SI: } \text{✓} )</td>
<td>( \text{SI: } \text{✓} )</td>
<td>( \text{SI: } \text{X} )</td>
</tr>
</tbody>
</table>

Fig. 1: Litmus tests illustrating transaction anomalies and their admissibility under SI and RSI. In all tests, initially, \( x = y = z = 0 \). The \( \#v \) annotation next to a read records the value read.
2.1 Towards an SI Reference Implementation without Timestamps

While the SI description above is suitable for understanding SI, it is not useful for integrating the SI model in a language such as C/C++ or Java. From a programmer’s perspective, in such languages the various threads directly access the uninstrumented (single-versioned) shared memory; they do not access their own instrumented snapshot at a particular timestamp, which is loosely related to the snapshots of other threads. Ideally, what we would therefore like is an equivalent description of SI in terms of accesses to uninstrumented shared memory and a synchronisation mechanism such as locks.

In what follows, we present our first lock-based reference implementation for SI that does not rely on timestamps. To do this, we assume that the locations accessed by a transaction can be statically determined. Specifically, we assume that each transaction $T$ is supplied with its read set, $RS$, and write set, $WS$, containing those locations read and written by $T$, respectively (a static over-approximation of these sets suffices for soundness.). As such, our first reference implementation is prescient [19] in that it requires a priori knowledge of the locations accessed by the transaction. Later in §2.3 we lift this assumption and develop an SI reference implementation that is non-prescient and similarly does not rely on timestamps.

Conceptually, a candidate implementation of transaction $T$ would (1) obtain a snapshot of the locations read by $T$; (2) lock those locations written by $T$; (3) execute $T$ locally; and (4) unlock the locations written. The snapshot is obtained via $\text{snapshot}(RS)$ in Fig. 3 where the values of locations in $RS$ are recorded in a local array $s$. The local execution of $T$ is carried out by executing $\langle T \rangle$ in Fig. 3, which is obtained from $T$ by (i) modifying read operations to read locally from the snapshot in $s$, and (ii) updating the snapshot after each write operation. Note that the snapshot must be obtained atomically to reflect the memory state at a particular instance (cf. start-timestamp). An obvious way to ensure the snapshot atomicity is to lock the locations in the read set, obtain a snapshot, and unlock the read set. However, as we must allow for two transactions reading from the same location to execute in parallel, we opt for multiple-readers-single-writer (MRSW) locks.

Let us now try to make this general pattern more precise. As a first attempt, consider the implementation in Fig. 2a written in a simple while language, which releases all the reader locks at the end of the snapshot phase before acquiring any writer locks. This implementation is unsound as it admits the lost update (LU) anomaly in Fig. 1 disallowed under SI [11]. To understand this, consider a scheduling where $T_2$ runs between lines 3 and 4 of $T_1$ in Fig. 2a, which would result in $T_1$ having read a stale value. The problem is that the writer locks on $WS$ are acquired too late, allowing two conflicting transactions to run concurrently. To address this, writer locks must be acquired early enough to pre-empt the concurrent execution of write-write-conflicting transactions. Note that locks have to be acquired early even for locations only written by a transaction to avoid exhibiting a variant of the lost update anomaly (LU2).
As such, our second candidate implementation in Fig. 2b brings forward the acquisition of writer locks. Whilst this implementation is sound (and disallows lost update), it nevertheless disallows behaviours deemed valid under SI such as the write skew anomaly (WS) in Fig. 1, and is thus incomplete. The problem is that such early acquisition of writer locks not only pre-empts concurrent execution of write-write-conflicting transactions, but also those of read-write-conflicting transactions (e.g. WS) due to the exclusivity of writer locks.

To remedy this, in our third candidate implementation in Fig. 2c we first acquire weaker reader locks on all locations in RS or WS, and later promote the reader locks on WS to exclusive writer ones, while releasing the reader locks on RS. The promotion of a reader lock signals its intent for exclusive ownership and awaits the release of the lock by other readers before claiming it exclusively as a writer. To avoid deadlocks, we further assume that RS ∪ WS is ordered so that locks are promoted in the same order by all threads.

Although this implementation is “more complete” than the previous one, it is still incomplete as it disallows certain behaviour admitted by SI. In particular, consider a variant of the write skew anomaly (WS2) depicted in Fig. 1, which is admitted under SI, but not admitted by this implementation.

To understand why this is admitted by SI, recall the operational SI model using timestamps. Let the domain of timestamps be that of natural numbers N. The behaviour of (WS2) can be achieved by assigning the following execution intervals for T1: \([t^1_0=2, t^1_c=2]\); T2: \([t^2_0=1, t^2_c=4]\); and T3: \([t^3_0=3, t^3_c=3]\). To see why the implementation in Fig. 2c does not admit the behaviour in (WS2), let us assume without loss of generality that x is ordered before y. Upon executing lines 3-5, a) T1 promotes y; b) T2 promotes x and then c) releases the reader lock on y; and d) T3 releases the reader lock on x. To admit the behaviour in (WS2), the release of y in (c) must occur before the promotion of y in (a) since otherwise T2 cannot read 0 for y. Similarly, the release of x in (d) must occur before its promotion in (b). On the other hand, since T3 is executed by the same thread after T1, we know that (a) occurs before (d). This however leads to circular execution: (b)→(c)→(a)→(d)→(b), which cannot be realised.
0. \( LS := \emptyset; \)
1. for \((x \in RS \cup WS)\) lock\(_r\) \(x\)
2. \(\text{snapshot}(RS);\)
3. for \((x \in RS\setminus WS)\) unlock\(_r\) \(x\)
4. for \((x \in WS)\) {
   5. if \((\text{can-promote} \ x)\) \(LS.add(x)\)
   6. else {
      7. for \((x \in LS)\) unlock\(_w\) \(x\)
      8. for \((x \in WS \setminus LS)\) unlock\(_r\) \(x\)
      9. goto line 0 }
   10. }
11. \((T);\)
12. for \((x \in WS)\) unlock\(_w\) \(x\)

\(\text{snapshot}(RS) \triangleq \text{for } (x \in RS) \ s_x := x\)

\(\text{snapshot}_{RSI}(RS) \triangleq\)
\(\text{\texttt{start}}:\) for \((x \in RS)\) \(s_x := x\)
\(\text{for } (x \in RS) \{\)
\(\text{if } (s_x! = x) \text{ goto start}\)
\} \(\text{\texttt{end}}\)
\(\{a := x\} \triangleq a := s_x\)
\(\{x := a\} \triangleq x := a; s_x := a\)
\(\langle S_1; S_2 \rangle \triangleq \langle S_1 \rangle; \langle S_2 \rangle\)
\(\langle \text{while}(e) \ S \rangle \triangleq \text{while}(e) \langle S \rangle\)
... and so on...

Fig. 3: SI implementation of transaction \(T\) given \(RS, WS\); the code in blue ensures deadlock avoidance. The RSI implementation (§5) is obtained by replacing \(\text{snapshot}\) on line 2 with \(\text{snapshot}_{RSI}\).

To overcome this, in our final candidate execution in Fig. 3 (ignoring the code in blue), after obtaining a snapshot, we \textit{first} release the reader locks on \(RS\), and then promote the reader locks on \(WS\), rather than simultaneously in one pass. As we demonstrate in §4, the implementation in Fig. 3 is both \textit{sound and complete} against its declarative SI specification.

**Avoiding Deadlocks** As two distinct reader locks on \(x\) may simultaneously attempt to promote their locks, promotion is done on a ‘first-come-first-served’ basis to avoid \textit{deadlocks}. A call to \texttt{can-promote} \(x\) by reader \(r\) thus returns a boolean denoting either (i) successful promotion (true); or (ii) failed promotion as another reader \(r'\) is currently promoting a lock on \(x\) (false). In the latter case, \(r\) must release its reader lock on \(x\) to ensure the successful promotion of \(x1\) by \(r'\) and thus avoid deadlocks. To this end, our implementation in Fig. 3 includes a deadlock avoidance mechanism (code in blue) as follows. We record a list \(LS\) of those locks on the write set that have been successfully promoted so far. When promoting a lock on \(x\) succeeds (line 5), the \(LS\) is extended with \(x\). On the other hand, when promoting \(x\) fails (line 6), all those locks promoted so far (i.e. in \(LS\)) as well as those yet to be promoted (i.e. in \(WS \setminus LS\)) are released and the transaction is restarted.

\textbf{Remark 1.} Note that the deadlock avoidance code in blue does not influence the correctness of the implementation in Fig. 3, and is merely included to make the reference implementation more realistic. In particular, the implementation without the deadlock avoidance code is both sound and complete against the SI specification, provided that the conditional \texttt{can-promote} call on line 5 is replaced by the blocking \texttt{promote} call.
Avoiding Over-Synchronisation due to MRSW Locks Consider the store buffering program (SBT) shown in Fig. 1. If, for a moment, we ignore transactional accesses, our underlying memory model (RA)—as well as all other weak memory models—allows the annotated weak behaviour. Intuitively, placing the two transactions that only read $z$ in (SBT) should still allow the weak behaviour since the two transactions do not need to synchronise in any way. Nevertheless, most MRSW lock implementations forbid this outcome because they use a single global counter to track the number of readers that have acquired the lock, which inadvertently also synchronises the readers with one another. As a result, the two read-only transactions act as memory fences forbidding the weak outcome of (SBT). To avoid such synchronisation, in the technical appendix [31] (Appendix A) we provide a different MRSW implementation using a separate location for each thread so that reader lock acquisitions do not synchronise.

To keep the presentation simple, we henceforth assume an abstract specification of a MRSW lock library providing operations for acquiring/releasing reader/writer locks, as well as promoting reader locks to writer ones. We require that (1) calls to writer locks (to acquire, release or promote) synchronise with all other calls to the lock library; and (2) writer locks provide mutual exclusion while held. We formalise these notions in §4. These requirements do not restrict synchronisation between two read lock calls: two read lock calls may or may not synchronise. Synchronisation between read lock calls is relevant only for the completeness of our RSI implementation (handling mixed-mode code); for that result, we further require that (3) read lock calls not synchronise.

2.2 Handling Racy Mixed-Mode Accesses

Let us consider what happens when data accessed by a transaction is modified concurrently by an uninstrumented atomic non-transactional write. Since such writes do not acquire any locks, the snapshots taken may include values written by non-transactional accesses. The result of the snapshot then depends on the order in which the variables are read. Consider the (MPT) example in Fig. 1. In our implementation, if in the snapshot phase $y$ is read before $x$, then the annotated weak behaviour is not possible because the underlying model (RA) disallows this weak “message passing” behaviour. If, however, $x$ is read before $y$, then the weak behaviour is possible. In essence, this means that the SI implementation described so far is of little use when there are races between transactional and non-transactional code. Technically, our SI implementation violates monotonicity with respect to wrapping code inside a transaction. The weak behaviour of the (MPT) example is disallowed by RA if we remove the transaction block T2, and yet it is exhibited by our SI implementation with the transaction block.

To get monotonicity under RA, it suffices for the snapshots to read the variables in the same order they are accessed by the transactions. Since a static calculation of this order is not always possible, following [30], we achieve this by reading each variable twice. In more detail, our snapshot$_{RSI}$ implementation in Fig. 3 takes two snapshots of the locations read by the transaction, and checks that they both return the same values for each location. This ensures
that every location is read both before and after every other location in the
transaction, and hence all the high-level happens-before orderings in executions
of the transactional program are also respected by its implementation. As we
demonstrate in §5, our RSI implementation is both sound and complete against
our proposed declarative semantics for RSI. There is however one caveat: since
equality of values is used to determine whether the two snapshots agree, we
will miss cases where different non-transactional writes to a location write the
same value. In our formal development (see §5), we thus assume that if multiple
non-transactional writes write the same value to the same location, they cannot
race with the same transaction. Note that this assumption cannot be lifted
without instrumenting non-transactional writes, and thus impeding performance
substantially. That is, to lift this restriction we must instead replace every
non-transactional write $x:= v$ with lock $w$ $x; x:= v; unlock w x$.

2.3 Non-Prescient Reference Implementations without Timestamps

Recall that the SI and RSI implementations in §2.1 are prescient in that they
require knowledge of the read and write sets of transactions beforehand. In what
follows we present alternative SI and RSI implementations that are non-prescient.

**Non-Prescient SI Reference Implementation** In Fig. 4 we present a lazy
lock-based reference implementation for SI. This implementation is non-prescient
and does not require a priori knowledge of the read set $RS$ and the write set $WS$.
Rather, the $RS$ and $WS$ are computed on the fly as the execution of the transaction
unfolds. As with the SI implementation in Fig. 3, this implementation does not
rely on timestamps and uses MRSW locks to synchronise concurrent accesses
to shared data. As before, the implementation consults a local snapshot at $s$
for read operations. However, unlike the eager implementation in Fig. 3 where
transactional writes are performed in-place, the implementation in Fig. 4 is lazy
in that it logs the writes in the local array $s$ and propagates them to memory at
commit time, as we describe shortly.

Ignoring the code in blue, the implementation in Fig. 4 proceeds with initial-
ising $RS$ and $WS$ with $\emptyset$ (line 1); it then populates the local snapshot array at $s$
with initial value $\perp$ for each location $x$ (line 2). It then executes $(T)$ which is
obtained from $T$ as follows. For each read operation $a:= x$ in $T$, first the value
of $s[x]$ is inspected to ensure it contains a snapshot of $x$. If this is not the case
(i.e. $x \not\in RS \cup WS$), a reader lock on $x$ is acquired, a snapshot of $x$ is recorded
in $s[x]$, and the read set $RS$ is extended with $x$. The snapshot value in $s[x]$ is
subsequently returned in $a$. Analogously, for each write operation $x:= a$, the $WS$
is extended with $x$, and the written value is lazily logged in $s[x]$. Recall from
our candidate executions in Fig. 2 that to ensure implementation correctness, for
each written location $x$, the implementation must first acquire a reader lock on $x$,
and subsequently promote it to a writer lock. As such, for each write operation
in $T$, the implementation first checks if a reader lock for $x$ has been acquired (i.e.$x \in RS \cup WS$) and obtains one if this is not the case.
0. \( LS := \emptyset \); 
1. \( RS := \emptyset \); \( WS := \emptyset \);
2. for (\( x \in \text{Locs} \)) \( s[x] := \perp \);
3. \( \{ T \} \);
4. for (\( x \in R_S \cup W_S \)) unlock_r x 
5. for (\( x \in W_S \)) 
6. if (can-promote \( x \)) \( LS.add(x) \)
7. else 
8. for (\( x \in L \)) unlock_w x
9. for (\( x \in W_S \setminus L \)) unlock_r x
10. goto line 0 
11. for (\( x \in W_S \)) \( x := s[x] \)
12. for (\( x \in W_S \)) unlock_w x

\( \{ a := x \} \triangleq \)
1. \( \{ x \not\in R_S \cup W_S \} \{ \)
2. lock_r \( x \); \( RS.add(x) \)
3. \( s[x] := x \);
4. \( } \}
5. \( a := s[x] \);

\( \{ x := a \} \triangleq \)
1. \( \{ x \not\in R_S \cup W_S \} \{ \)
2. lock_r \( x \); \( WS.add(x) \); \( s[x] := a \);
3. \( } \}
4. \( \{ S_1 ; S_2 \} \triangleq \{ S_1 \} ; \{ S_2 \} \)

\( \{ \text{while} (e) S \} \triangleq \text{while} (e) \{ S \} \)

\( \ldots \text{and so on...} \)

Fig. 4: Non-prescient SI implementation of transaction T with RS and WS computed on the fly; the code in blue ensures deadlock avoidance.

Once the execution of \( \{ T \} \) is completed, the implementation proceeds to commit the transaction. To this end, the reader locks on RS are released (line 4), reader locks on WS are promoted to writer ones (line 6), the writes logged in \( s \) are propagated to memory (line 11), and finally the writer locks on WS are released (line 12). As we demonstrate later in §4, the implementation in Fig. 4 is both sound and complete against the declarative SI specification.

Note that the implementation in Fig. 4 is optimistic in that it logs the writes performed by the transaction in the local array \( s \) and propagates them to memory at commit time, rather than performing the writes in-place as with its pessimistic counterpart in Fig. 3. As before, the code in blue ensures deadlock avoidance and is identical to its counterpart in Fig. 3. As before, this deadlock avoidance code does not influence the correctness of the implementation and is merely included to make the reference implementation more practical.

Non-Prescient RSI Reference Implementation In Fig. 5 we present a lazy lock-based reference implementation for RSI. As with its SI counterpart, this implementation is non-prescient and computes the RS and WS on the fly. As before, the implementation does not rely on timestamps and uses MRSW locks to synchronise concurrent accesses to shared data. Similarly, the implementation consults the local snapshot at \( s \) for read operations, whilst logging write operations lazily in a write sequence at \( wseq \), as we describe shortly.

Recall from the RSI implementation in §2.1 that to ensure snapshot validity, each location is read twice to preclude intermediate non-transactional writes. As such, when writing to a location \( x \), the initial value read (recorded in \( s \)) must not be overwritten by the transaction to allow for subsequent validation of the snapshot. To this end, for each location \( x \), the snapshot array \( s \) contains a pair of values, \((r,c)\), where \( r \) denotes the snapshot value (initial value read), and \( c \) denotes the current value which may have overwritten the snapshot value.

Recall that under weak isolation, the intermediate values written by a transaction may be observed by non-transactional reads. For instance, given the
Fig. 5: Non-prescient RSI implementation of transaction \( T \) with \( RS \) and \( WS \) computed on the fly; the code in blue ensures deadlock avoidance.

**T:** \( \{ x := 1 ; x := 2 \mid a := x \} \) program, the non-transactional read \( a := x \), may read either 1 or 2 for \( x \). As such, at commit time, it is not sufficient solely to propagate the last written value (in program order) to each location (e.g. to propagate only the \( x := 2 \) write in the example above). Rather, to ensure implementation completeness, one must propagate all written values to memory, in the order they appear in the transaction body. To this end, we track the values written by the transaction as a (FIFO) write sequence at location \( wseq \), containing items of the form \((x,v)\), denoting the location written \((x)\) and the associated value \((v)\).

Ignoring the code in blue, the implementation in Fig. 5 initialises \( RS \) and \( WS \) with \( \emptyset \), initialises \( wseq \) as an empty sequence \([\] \) (line 1), and populates the local snapshot array \( s \) with initial value \((\perp, \perp)\) for each location \( x \) (line 2). It then executes \( (T) \) obtained from \( T \) in an analogous manner to that in Fig. 4. For every read \( a := x \) in \( (T) \), the current value recorded for \( x \) in \( s \) (namely \( c \) when \( s[x] \) holds \((\perp,c)\)) is returned in \( a \). Dually, for every write \( x := a \) in \( (T) \), the current value recorded for \( x \) in \( s \) is updated to \( a \), and the write is logged in the write sequence \( wseq \) by appending \((x,a)\) to it.

Upon completion of \( (T) \), the snapshot in \( s \) is validated (lines 4-7). Each location \( x \) in \( RS \) is thus read again and its value is compared against the snapshot value in \( s[x] \). If validation fails (line 5), the locks acquired are released (line 6) and the transaction is restarted (line 7).

If validation succeeds, the transaction is committed: the reader locks on \( RS \) are released (line 8), the reader locks on \( WS \) are promoted (line 10), the writes in \( wseq \) are propagated to memory in FIFO order (line 15), and finally the writer locks on \( WS \) are released (line 16).

As we show in §5, the implementation in Fig. 5 is both sound and complete against our proposed declarative specification for RSI. As before, the code in blue ensures deadlock avoidance; it does not influence the implementation correctness and is merely included to make the implementation more practical.
Supporting Explicit Abort Instructions It is straightforward to extend the lazy implementations in Fig. 4 and Fig. 5 to handle transactions containing explicit abort instructions. More concretely, as the effects (writes) of a transaction are logged locally and are not propagated to memory until commit time, upon reaching an abort in $\{T\}$ no roll-back is necessary, and one can simply release the locks acquired so far and return. That is, one can extend $\{\}$ in Fig. 4 and Fig. 5, and define $(\text{abort}) \triangleq \text{for } (x \in \text{RS} \cup \text{WS}) \text{unlock}_x x; \text{return}$. 

3 A Declarative Framework for STM

We present the notational conventions used in the remainder of this article, and describe a general framework for declarative concurrency models. Later in this article, we present SI, its extension with non-transactional accesses, and their lock-based implementations as instances of this general definition.

Notation Given a relation $r$ on a set $A$, we write $r^+$, $r^*$ and $r^-$ for the reflexive, transitive and reflexive-transitive closure of $r$, respectively. We write $r^−$ for the inverse of $r$; $r_A$ for $r \cap (A \times A)$; $|A|$ for the identity relation on $A$, i.e. $\{(a, a) | a \in A\}$: $\text{irreflexive}(r)$ for $\nexists a. (a, a) \in r$; and $\text{acyclic}(r)$ for $\text{irreflexive}(r^+)$. Given two relations $r_1$ and $r_2$, we write $r_1; r_2$ for their (left) relational composition, i.e. $\{(a, b) | \exists c. (a, c) \in r_1 \land (c, b) \in r_2\}$. Lastly, when $r$ is a strict partial order, we write $r|_{\text{imm}}$ for the immediate edges in $r$: $\{(a, b) \in r | \nexists c. (a, c) \in r \land (c, b) \in r\}$.

Assume finite sets of locations $\text{Loc}$; values $\text{Val}$; thread identifiers $\text{TId}$, and transaction identifiers $\text{TXId}$. We use $x, y, z$ to range over locations, $v$ over values, $\tau$ over thread identifiers, and $\xi$ over transaction identifiers.

Definition 1 (Events). An event is a tuple $(n, \tau, \xi, l)$, where $n \in \mathbb{N}$ is an event identifier, $\tau \in \text{TId} \cup \{0\}$ is a thread identifier (0 is used for initialisation events), $\xi \in \text{TXId} \cup \{0\}$ is a transaction identifier (0 is used for non-transactional events), and $l$ is an event label that takes one of the following forms:

- A memory access label: $\text{R}(x, v)$ for reads; $\text{W}(x, v)$ for writes; and $\text{U}(x, v_r, v_w)$ for updates.
- A lock label: $\text{RL}(x)$ for reader lock acquisition; $\text{RU}(x)$ for reader lock release; $\text{WL}(x)$ for writer lock acquisition; $\text{WU}(x)$ for writer lock release; and $\text{PL}(x)$ for reader to writer lock promotion.

We typically use $a$, $b$, and $e$ to range over events. The functions $\text{tid}$, $\text{tx}$, $\text{lab}$, $\text{typ}$, $\text{loc}$, $\text{val}_x$ and $\text{val}_v$ respectively project the thread identifier, transaction identifier, label, type (in $\{\text{R}, \text{W}, \text{U}, \text{RL}, \text{RU}, \text{WL}, \text{WU}, \text{PL}\}$), location, and read/written values of an event, where applicable. We assume only reads and writes are used in transactions $(\text{tx}(a) \neq 0 \implies \text{typ}(a) \in \{\text{R}, \text{W}\})$.

Given a relation $r$ on events, we write $r_{\text{loc}}$ for $\{(a, b) \in r | \text{loc}(a) = \text{loc}(b)\}$. Analogously, given a set $A$ of events, we write $A_x$ for $\{a \in A | \text{loc}(a) = x\}$.

Definition 2 (Execution graphs). An execution graph, $G$, is a tuple of the form $(E, \text{po}, \text{rf}, \text{mo}, \text{lo})$, where:
- $E$ is a set of events, assumed to contain a set $E_0$ of initialisation events, consisting of a write event with label $\xi w(x,0)$ for every $x \in \text{Loc}$. The sets of read events in $E$ is denoted by $\mathcal{R} \triangleq \{ e \in E \mid \text{typ}(e) \in \{R,U\} \}$; write events by $\mathcal{W} \triangleq \{ e \in E \mid \text{typ}(e) \in \{\mathcal{W},\mathcal{U}\} \}$; update events by $\mathcal{U} \triangleq \mathcal{R} \cap \mathcal{W}$; and lock events by $\mathcal{L} \triangleq \{ e \in E \mid \text{typ}(e) \in \{\mathcal{L},\mathcal{R},\mathcal{W},\mathcal{U},\mathcal{P}\} \}$. The sets of reader lock acquisition and release events, $\mathcal{RL}$ and $\mathcal{RU}$, writer lock acquisition and release events, $\mathcal{WL}$ and $\mathcal{WU}$, and lock promotion events $\mathcal{PL}$ are defined analogously. The set of transactional events in $E$ is denoted by $\mathcal{N} \mathcal{T} \triangleq \{ e \in E \mid \text{tx}(e) \neq 0 \}$; and the set of non-transactional events is denoted by $\mathcal{N} \mathcal{N} \mathcal{T} \triangleq E \setminus \mathcal{N} \mathcal{T}$.

- $\text{po} \subseteq E \times E$ denotes the ‘program-order’ relation, defined as a disjoint union of strict total orders, each ordering the events of one thread, together with $E_0 \times (E \setminus E_0)$ that places the initialisation events before any other event. We assume that events belonging to the same transaction are ordered by $\text{po}$, and that any other event po-between them also belongs to the same transaction.

- $\text{rf} \subseteq \mathcal{W} \times \mathcal{R}$ denotes the ‘reads-from’ relation, defined between write and read events of the same location with matching read and written values; it is total and functional on reads, i.e. every read is related to exactly one write.

- $\text{mo} \subseteq \mathcal{W} \times \mathcal{W}$ denotes the ‘modification-order’ relation, defined as a disjoint union of strict total orders, each ordering the write events on one location.

- $\text{lo} \subseteq \mathcal{L} \times \mathcal{L}$ denotes the ‘lock-order’ relation, defined as a disjoint union of strict orders, each of which (partially) ordering the lock events to one location.

In the context of an execution graph $G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo})$—we often use “$G$” as a prefix to make this explicit—the ‘same-transaction’ relation, $\text{st} \in \mathcal{T} \times \mathcal{T}$, is the equivalence relation given by $\text{st} \triangleq \{(a,b) \in \mathcal{T} \times \mathcal{T} \mid \text{tx}(a) = \text{tx}(b)\}$. Given a relation $r \subseteq E \times E$, we write $r_T$ for lifting $r$ to transaction classes: $r_T \triangleq \text{st}; (r \setminus \text{st}); \text{st}$. For instance, when $(w, r) \in \text{rf}$, $w$ is a transaction $\xi_1$ event and $r$ is a transaction $\xi_2$ event, then all events in $\xi_1$ are $\text{rf}_T$-related to all events in $\xi_2$. Analogously, we write $r_I$ to restrict $r$ to its intra-transactional edges (within a transaction): $r_I \triangleq r \cap \text{st}$; and write $r_E$ to restrict $r$ to its extra-transactional edges (outside a transaction): $r_E \triangleq r \setminus \text{st}$. Lastly, the ‘reads-before’ relation is defined by $\text{rb} \triangleq (\text{rf}^{-1}; \text{mo}) \setminus [E]$. Intuitively, rb relates a read $r$ to all writes $w$ that are mo-after the write $r$ reads from; i.e. when $(w', r) \in \text{rf}$ and $(w', w) \in \text{mo}$, then $(r, w) \in \text{rb}$.

In the transactional literature, this is known as the anti-dependency relation [4,3].

Execution graphs of a given program represent traces of shared memory accesses generated by the program. The set of execution graphs associated with programs written in our while language can be straightforwardly defined by induction over the structure of programs as in e.g. [35]. Each execution of a program $P$ has a particular program outcome, prescribing the final values of local variables in each thread. In this initial stage, the execution outcomes are almost unrestricted as there are very few constraints on the $\text{rf}$, $\text{mo}$ and $\text{lo}$ relations. Such restrictions and thus the permitted outcomes of a program are determined by defining the set of consistent executions, which is defined separately for each model we consider. Given a program $P$ and a model $M$, the set $\text{outcomes}_M(P)$ collects the outcomes of every $M$-consistent execution of $P$. 
4 Snapshot Isolation (SI)

We present a declarative specification of SI and demonstrate that the SI implementations presented in Fig. 3 and Fig. 4 are both sound and complete with respect to the SI specification.

In [11] Cerone and Gotsman developed a declarative specification for SI using dependency graphs [4,3]. Below we adapt their specification to the notation of §3. As with [11], throughout this section, we take SI execution graphs to be those in which $E = T \subseteq (R \cup W) \setminus U$. That is, the SI model handles transactional code only, consisting solely of read and write events (excluding updates).

Definition 3 (SI consistency [11]). An SI execution $G = (E, po, rf, mo, lo)$ is SI-consistent if the following conditions hold:

- $rf_I \cup mo_I \cup rb_I \subseteq po$ (INT)
- acyclic($po_T \cup rf_T \cup mo_T; rb_T)$ (EXT)

Informally, (INT) ensures the consistency of each transaction internally, while (EXT) provides the synchronisation guarantees among transactions. In particular, we note that the two conditions together ensure that if two read events in the same transaction read from the same location $x$, and no write to $x$ is po-between them, then they must read from the same write (known as ‘internal read consistency’).

Next, we provide an alternative formulation of SI-consistency which will serve as the basis of our extension with non-transactional accesses in §5. In the technical appendix [31] we prove that the two formulations are equivalent.

Proposition 1. An SI execution $G = (E, po, rf, mo, lo)$ is SI-consistent if and only if INT holds and the ‘SI-happens-before’ relation $si-hb \triangleq (po_T \cup rf_T \cup mo_T \cup si-rb)^+$ is irreflexive, where $si-rb \triangleq [R_E]; rb_T; |W|$ and $R_E \triangleq \{r \mid \exists w. (w, r) \in rf_E\}$.

Proof. The full proof is given in the technical appendix [31] (see Prop. 2 in Appendix B).

Intuitively, SI-happens-before orders events of different transactions; this order is due to either the program order ($po_T$), or synchronisation enforced by the implementation ($rf_T \cup mo_T \cup si-rb$). By contrast, events of the same transaction are unordered, as the implementation may well execute them in a different order (in particular, by taking a snapshot, it executes external reads before the writes).

In more detail, the $rf_T$ corresponds to transactional synchronisation due to causality, i.e. when one transaction $T_2$ observes an effect of an earlier transaction $T_1$. The inclusion of $rf_T$ ensures that $T_2$ cannot read from $T_1$ without observing its entire effect. This in turn ensures that transactions exhibit ‘all-or-nothing’ behaviour: they cannot mix-and-match the values they read. For instance, if $T_1$ writes to both $x$ and $y$, transaction $T_2$ may not read $x$ from $T_1$ but read $y$ from an earlier (in ‘happens-before’ order) transaction $T_0$.

The $mo_T$ corresponds to transactional synchronisation due to write-write conflicts. Its inclusion enforces write-conflict-freedom of SI transactions: if $T_1$ and
\( T_2 \) both write to \( x \) via events \( w_1 \) and \( w_2 \) such that \((w_1, w_2) \in \text{mo} \), then \( T_1 \) must commit before \( T_2 \), and thus its entire effect must be visible to \( T_2 \).

To understand \( \text{si-rb} \), first note that \( \mathcal{R}_E \) denotes the \textit{external} transactional reads (i.e. those reading a value written by another transaction). That is, the \( \mathcal{R}_E \) are the read events that get their values from the transactional snapshot phases. By contrast, internal reads (those reading a value written by the same transaction) synchronise write mode by another thread until it has been released by its current owner.

We assume that locks in implementation programs are used in a \textit{well-formed} manner: the sequence of lock events for \( T \) is recorded. Similarly, there exists a commit-timestamp \( t^*_c \) as of which the updates of \( T \) are committed. Moreover, since \((r, w) \in \text{rb}\) we know \( t^*_0 < t^*_c \) (otherwise \( r \) must read the value written by \( w \) and not \( w' \)). That is, we know all events in the snapshot of \( T_1 \) (i.e. all external reads in \( T_1 \)) happen before all writes of \( T_2 \).

We use the declarative framework in §3 to formalise the semantics of our implementation. Here, our programs include only non-transactional code, and thus implementation execution graphs are taken as those in which \( T = \emptyset \). Furthermore, we assume that locks in implementation programs are used in a \textit{well-formed} manner: the sequence of lock events for each location, in each thread (following po), should match (a prefix of) the regular expression \((\text{RL} \cdot \text{RU} | \text{WL} \cdot \text{WU} | \text{RL} \cdot \text{PL} \cdot \text{WU})^*\).

For instance, a thread never releases a lock, without having acquired it earlier in the program. As a consistency predicate on execution graphs, we use the C11 release/acquire consistency augmented with certain constraints on lock events.

\textbf{Definition 4.} An implementation execution graph \( G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo}) \) is RA-consistent if the following hold, where \( \text{hb} \triangleq (\text{po} \cup \text{rf} \cup \text{lo})^+ \) denotes the \textquote{RA-happens-before} relation:

\[
\begin{align*}
- \quad & \forall x; \forall a \in \text{WL}_x \cup \text{WU}_x \cup \mathcal{PL}_x; \quad b \in \mathcal{L}_x; \quad a = b \lor (a, b) \in \text{lo} \lor (b, a) \in \text{lo} \quad \text{(WSync)} \\
- \quad & [\text{WL} \cup \mathcal{PL}]; [\text{lo} \setminus \text{po}]; [\mathcal{L}] \subseteq \text{po}; [\text{WU}]; \text{lo} \quad \text{(WEX)} \\
- \quad & [\mathcal{RL}]; (\text{lo} \setminus \text{po}); [\text{WL} \cup \mathcal{PL}] \subseteq \text{po}; [\mathcal{RU} \cup \mathcal{PL}]; \text{lo} \quad \text{(RSshare)} \\
- \quad & \text{acyclic}(\text{hb}_{\text{loc}} \cup \text{mo} \cup \text{rb}) \quad \text{(ACYC)}
\end{align*}
\]

The (WSync) states that write lock calls (to acquire, release or promote) synchronise with all other calls to the same lock.

The next two constraints ensure the \textit{single-writer-multiple-readers} paradigm. In particular, (WEX) states that write locks provide mutual exclusion while held: any lock event \( l \) of thread \( \tau \) \text{lo}-after a write lock acquisition or promotion event \( l' \) of another thread \( \tau' \), is \text{lo}-after a subsequent write lock release event \( u \) of \( \tau' \) (i.e. \((l', u) \in \text{po}\) and \((u, l) \in \text{lo}\)). As such, the lock cannot be acquired (in read or write mode) by another thread until it has been released by its current owner.

\footnote{By taking \text{rbr} instead of \text{si-rb} in Prop. 1 one obtains a characterisation of serialisability.}
The \text{(RShare)} analogously states that once a thread acquires a lock in read mode, the lock cannot be acquired in write mode by other threads until it has either been released, or promoted to a writer lock (and subsequently released) by its owner. Note that this does not preclude other threads from simultaneously acquiring the lock in read mode. In the technical appendix \cite{31} we present two MRSW lock implementations that satisfy the conditions outlined above.

The last constraint \text{(Acyc)} is that of C11 RA consistency \cite{23}, with the \text{hb} relation extended with \text{lo}.

Remark 2. Our choice of implementing the SI STMs on top of the RA fragment is purely for presentational convenience. Indeed, it is easy to observe that execution graphs of \(LPM\) are data race free, and thus, \text{Acyc} could be replaced by any condition that implies \(\forall x. ([W_x]; (po \cup lo)^+; [W_x]; (po \cup lo)^+; [R_x]) \cap rf = \emptyset\) and that is implied by \text{acyclic}(po \cup rf \cup lo \cup mo \cup rb). In particular, the C11 non-atomic accesses or sequentially consistent accesses may be used.

We next show that our SI implementations in Fig. 3 and Fig. 4 are sound and complete with respect to the declarative specification given above. The proofs are non-trivial and the full proofs are given in the technical appendix \cite{31}.

\textbf{Theorem 1 (Soundness and completeness).} Let \(P\) be a transactional program; let \(LPM_x\) denote its eager implementation as given in Fig. 3 and \(LPM_l\) denote its lazy implementation as given in Fig. 4. Then:

\[
\text{outcomes}_{SI}(P) = \text{outcomes}_{RA}(LPM_x) = \text{outcomes}_{RA}(LPM_l)
\]

\textbf{Proof.} The full proofs for both implementations is given in the technical appendix \cite{31} (Appendix C and Appendix D, respectively).

\textbf{Stronger MRSW Locks} As noted in \S 2, for both (prescient and non-prescient) SI implementations our soundness and completeness proofs show that the same result holds for a stronger lock specification, in which reader locks synchronise as well. Formally, this specification is obtained by adding the following to Def. 4:

\[
\forall x. \forall a, b \in RL_x \cup RL_x. a = b \lor (a, b) \in lo \lor (b, a) \in lo \quad (\text{RSync})
\]

Soundness of this stronger specification (\(\text{outcomes}_{RA}(LPM_x) \subseteq \text{outcomes}_{SI}(P)\) for \(x \in \{e, l\}\)) follows immediately from Thm. 1.Completeness (\(\text{outcomes}_{SI}(P) \subseteq \text{outcomes}_{RA}(LPM_x)\) for \(x \in \{e, l\}\)), however, is more subtle, as we need to additionally satisfy \text{(RSync)} when constructing \text{lo}. While we can do so for SI, it is essential for the completeness of our RSI implementations that reader locks not synchronise, as shown by \text{(SBT)} in \S 2.

In the technical appendix \cite{31} we present two MRSW lock implementations (see Appendix A). Both implementations are \textit{sound} against the \text{lo} conditions in Def. 4. Additionally, the first implementation is \textit{complete} against the conditions of Def. 4 augmented with \text{(RSync)}, whilst the second is complete against the conditions of Def. 4 alone.
5 Robust Snapshot Isolation (RSI)

We explore the semantics of SI STMs in the presence of non-transactional code with weak isolation guarantees (see §2). We refer to this model as robust snapshot isolation (RSI), due to its ability to provide SI guarantees between transactions even in the presence of non-transactional code. We propose the first declarative specification of RSI programs and develop two lock-based reference implementations that are both sound and complete against our proposed specification.

A Declarative Specification of RSI STMs We formulate a declarative specification of RSI semantics by adapting the SI semantics in Prop. 1 to account for non-transactional accesses. To specify the abstract behaviour of RSI programs, RSI execution graphs are taken to be those in which $L = \emptyset$. Moreover, as with SI graphs, RSI execution graphs are those in which $T \subseteq (R \cup W) \setminus U$. That is, RSI transactions comprise solely read and write events, excluding updates.

Definition 5 (RSI consistency). An execution $G = (E, po, rf, mo, lo)$ is RSI-consistent iff INT holds and acyclic($rsi-hb_{loc} \cup mo \cup rb$), where $rsi-hb \triangleq (rsi-po \cup rsi-rf \cup mo_T \cup si-rb)^+$ is the ‘RSI-happens-before’ relation, with $rsi-po \triangleq (po \setminus po_I) \cup \{W\} \cup \{W\}$ and $rsi-rf \triangleq (rf; [\mathcal{N}^T]) \cup ([\mathcal{N}^T]; rf; st) \cup rf_T \cup (mo; rf)_T$.

As with SI and RA, we characterise the set of executions admitted by RSI as graphs that lack cycles of certain shapes. To account for non-transactional accesses, similar to RA, we require $rsi-hb_{loc} \cup mo \cup rb$ to be acyclic (recall that $rsi-hb_{loc} \triangleq \{(a, b) \in rsi-hb \mid 1oc(a) = 1oc(b)\}$). The RSI-happens-before relation $rsi-hb$ includes both the synchronisation edges enforced by the transactional implementation (as in $si-hb$), and those due to non-transactional accesses (as in $hb$ of the RA consistency). The $rsi-hb$ relation itself is rather similar to $si-hb$. In particular, the $mo_T$ and $si-rb$ subparts can be justified as in $si-hb$; the difference between the two lies in $rsi-po$ and $rsi-rf$.

To justify $rsi-po$, recall from §4 that $si-hb$ includes $po_T$. The $rsi-po$ is indeed a strengthening of $po_T$ to account for non-transactional events: it additionally includes (i) $po$ to and from non-transactional events; and (ii) $po$ between two write events in a transaction. We believe (i) comes as no surprise to the reader;
for (ii), consider the execution graph in Fig. 6a, where transaction T is denoted by the dashed box labelled T, comprising the write events w₁ and w₂. Removing the T block (with w₁ and w₂ as non-transactional writes), this execution is deemed inconsistent, as this weak “message passing” behaviour is disallowed in the RA model. We argue that the analogous transactional behaviour in Fig. 6a must be similarly disallowed to maintain monotonicity with respect to wrapping non-transactional code in a transaction (see Thm. 3). As in SI, we cannot include the entire po in rsi-hb because the write-read order in transactions is not preserved by the implementation.

Similarly, rsi-rf is a strengthening of rfₜ to account for non-transactional events: in the absence of non-transactional events rsi-rf reduces to rfₜ ∪ (mo; rf)ₜ which is contained in si-hb. The rf; [NT] part is required to preserve the ‘happens-before’ relation for non-transactional code. That is, as rf is included in the hb relation of underlying memory model (RA), it is also included in rsi-hb.

The [NT]; rf; st part asserts that in an execution where a read event r of transaction T reads from a non-transactional write w, the snapshot of T reads from w and so all events of T happen after w. Thus, in Fig. 6b, r' cannot read from the overwritten initialisation write to y.

For the (mo; rf)ₜ part, consider the execution graph in Fig. 6c where there is a write event w of transaction T₁ and a read event r of transaction T₂ such that (w, r) ∈ mo; rf. Then, transaction T₂ must acquire the read lock of loc(w) after T₁ releases the writer lock, which in turn means that every event of T₁ happens before every event of T₂.

Remark 3. Recall that our choice of modelling SI and RSI STMs in the RA fragment is purely for presentational convenience (see Remark 2). Had we chosen a different model, the RSI consistency definition (Def. 5) would largely remain unchanged, with the exception of rsi-rf ≜ (sw; [NT]) ∪ ([NT]; sw; st) ∪ rfₜ ∪ (mo; rf)ₜ, where in the highlighted changes the rf relation is replaced with sw, denoting the ‘synchronises-with’ relation. As in the RA model sw ≜ rf, we have inlined this in Def. 5.

SI and RSI Consistency We next demonstrate that in the absence of non-transactional code, the definitions of SI-consistency (Prop. 1) and RSI-consistency (Def. 5) coincide. That is, for all executions G, if G.NT = ∅, then G is SI-consistent if and only if G is RSI-consistent. This is captured in the following theorem with its full proof given in the technical appendix [31].

Theorem 2. For all executions G, if G.NT = ∅, then:

\[ G \text{ is SI-consistent } \iff G \text{ is RSI-consistent} \]

Proof. The full proof is given in the technical appendix [31] (see Thm. 5 in Appendix B).

Note that the above theorem implies that for all transactional programs P, if P contains no non-transactional accesses, then outcomesₜ(P) = outcomesᵦₛ(P).
RSI Monotonicity

We next prove the monotonicity of RSI when wrapping non-transactional events into a transaction. That is, wrapping a block of non-transactional code inside a new transaction does not introduce additional behaviours. More concretely, given a program \( P \), when a block of non-transactional code in \( P \) is wrapped inside a new transaction to obtain a new program \( P_T \), then \( \text{outcomes}_{\text{RSI}}(P_T) \subseteq \text{outcomes}_{\text{RSI}}(P) \). This is captured in the theorem below, with its full proof given in the technical appendix [31].

**Theorem 3 (Monotonicity).** Let \( P_T \) and \( P \) be RSI programs such that \( P_T \) is obtained from \( P \) by wrapping a block of non-transactional code inside a new transaction. Then:

\[
\text{outcomes}_{\text{RSI}}(P_T) \subseteq \text{outcomes}_{\text{RSI}}(P)
\]

**Proof.** The full proof is given in the technical appendix [31] (see Thm. 6 in Appendix B).

Lastly, we show that our RSI implementations in §2 (Fig. 3 and Fig. 5) are sound and complete with respect to Def. 5. This is captured in the theorem below. The soundness and completeness proofs are non-trivial; the full proofs are given in the technical appendix [31].

**Theorem 4 (Soundness and completeness).** Let \( P \) be a program that possibly mixes transactional and non-transactional code. Let \( (P)_E \) denote its eager RSI implementation as given in Fig. 3 and \( (P)_L \) denote its lazy RSI implementation as given in Fig. 5.

If for every location \( x \) and value \( v \), every RSI-consistent execution of \( P \) contains either (i) at most one non-transactional write of \( v \) to \( x \); or (ii) all non-transactional writes of \( v \) to \( x \) are happens-before-ordered with respect to all transactions accessing \( x \), then:

\[
\text{outcomes}_{\text{RSI}}(P) = \text{outcomes}_{\text{RA}}((P)_E) = \text{outcomes}_{\text{RA}}((P)_L)
\]

**Proof.** The full proofs for both implementations is given in the technical appendix [31] (Appendix E and Appendix F, respectively).

6 Related and Future Work

Much work has been done in formalising the semantics of weakly consistent database transactions [7,34,3,4,10,12,11,13,14,18], both operationally and declaratively. On the operational side, Berenson et al. [7] gave an operational model of SI as a multi-version concurrent algorithm. Later, Sovran et al. [34] described and operationally defined the parallel snapshot isolation model (PSI), as a close relative of SI with weaker guarantees.

On the declarative side, Adya et al. [3,4] introduced dependency graphs (similar to execution graphs of our framework in §3) for specifying transactional semantics and formalised several ANSI isolation levels. Cerone et al. [10,12] introduced
abstract executions and formalised several isolation levels including SI and PSI. Later in [11], they used dependency graphs of Adya to develop equivalent SI and PSI semantics; recently in [13], they provided a set of algebraic laws for connecting these two declarative styles.

To facilitate client-side reasoning about the behaviour of database transactions, Gotsman et al. [18] developed a proof rule for proving invariants of client applications under a number of consistency models.

Recently, Kaki et al. [21] developed a program logic to reason about transactions under ANSI SQL isolation levels (including SI). To do this, they formulated an operational model of such programs (parametric in the isolation level). They then proved the soundness of their logic with respect to their proposed operational model. However, the authors did not establish the soundness or completeness of their operational model against existing formal semantics, e.g. [11]. The lack of the completeness result means that their proposed operational model may exclude behaviours deemed valid by the corresponding declarative models. This is a particular limitation as possibly many valid behaviours cannot be shown correct using the logic and is thus detrimental to its usability.

By contrast, the semantics of transactions in the STM setting with mixed-mode (both transactional and non-transactional) accesses is under-explored on both operational and declarative sides. Recently, Dongol at al. [17] applied execution graphs [5] to specify the behaviour of serialisable STM programs under weak memory models. Raad et al. [30] formalised the semantics of PSI STMs both declaratively (using execution graphs) and operationally (as lock-based reference implementations). Neither work, however, handles the semantics of SI STMs under weaker isolation guarantees.

Finally, Khyzha et al. [22] formalise the sufficient conditions on STMs and the programs running on them that together ensure strong isolation. That is, non-transactional accesses can be viewed as singleton transactions (transactions containing single instructions). However, their conditions require serialisability for fully transactional programs, and as such, RSI transactions do not meet their conditions. Nevertheless, we conjecture that a DRF guarantee for strong atomicity, similar to the one in [22], may be established for RSI. That is, if all executions of a given fully transactional program have no races between singleton and non-singleton transactions, then it is safe to replace all singleton transactions by non-transactional accesses.

In the future, we plan to build on the work presented here by developing reasoning techniques that would allow us to verify properties of STM programs. This can be achieved by either extending existing program logics for weak memory, or developing new ones for currently unsupported models. In particular, we can reason about the SI models presented here by developing custom proof rules in the existing program logics for RA such as [24,35].

Acknowledgements We thank the VMCAI reviewers for their constructive feedback. The first author was supported in part by a European Research Council (ERC) Consolidator Grant for the project “RustBelt”, under the European Union’s Horizon 2020 Framework Programme (grant agreement number 683289).
References


A MRSW Lock Implementations

We consider two different MRSW library implementations, both satisfying the lock synchronisation guarantees required by Def. 4. As we demonstrate shortly, our first implementation (Appendix A.1) offers additional synchronisation guarantees by ensuring that any two calls to the library (including those of read locks) synchronise. That is, our first implementation satisfies the (RSync) axiom on page 16. By contrast, the synchronisation guarantees of our second implementation (Appendix A.2) are exactly those required by Def. 4, where a library call to a write lock synchronises with all other lock library calls. As we discussed earlier in §2, whilst both library implementations can be used in our SI implementation, only the weaker (second) implementation can be used in our RSI implementation.

A.1 Fully Synchronising MRSW Lock Implementation

Our first MRSW lock library is implemented in the RA fragment of C11 [23], and is given in Fig. 7. In this implementation, the lock associated with each location \( x \) resides at location \( x+1 \), written \( x_l \). The state of a lock \( x_l \) is represented by an integer value. A lock \( x_l \) may hold either:

i) value 0, denoting that the lock is free (not held in read or write mode); or
ii) value 1, denoting that the lock is held (exclusively) in write mode; or
iii) an even value \( 2^n \) with \( n > 0 \), denoting that the lock is held in (shared) read mode by \( n \) readers; or
iv) an odd value \( 2^n+1 \) with \( n > 0 \), denoting that the lock is currently being promoted, awaiting the release of \( n \) readers.

As such, the implementation of \( \text{lock}_w \ x \) simply spins until it can atomically update (via \( \text{CAS} \)) the value of \( x_l \) from zero (free) to one (acquired in write mode). The \( \text{CAS}(x_l, v, v') \) denotes the atomic ‘compare-and-set’ operation, where either \( x_l \) currently holds value \( v \) in which case it is atomically updated to \( v' \) and true is returned; or \( x_l \) currently holds a value other than \( v \) in which case it is left unchanged and false is returned. Dually, the implementation of \( \text{unlock}_w \ x \) simply releases the write lock by atomically assigning \( x_l \) to zero.

The implementation of \( \text{can-promote} \ x \) is more involved. As multiple readers may attempt to promote their reader locks simultaneously, promotion is granted on a ‘first-come-first-served’ bases. As such, the implementation of \( \text{can-promote} \ x \) first reads the value of \( x_l \). If \( x_l \) holds an odd value, then another reader is currently promoting \( x_l \) and thus promotion of \( x_l \) fails by returning false. On the other hand, if \( x_l \) holds an even value, then its value is atomically decremented (to an odd value) to signal the intention to promote. The implementation then proceeds by spinning until all other readers have released their locks on \( x \) (i.e. \( x_l == 1 \)), at which point true is returned to denote the successful acquisition of \( x \) in write mode. Note that once a reader has signalled its intention to promote \( x \) (by decrementing \( x_l \) to an odd value), any other such attempt to promote the lock on \( x \), as well as calls to acquire it in read mode will fail thereafter until such time that \( x \) is released by its current promoter.
The implementation of $\text{lock}_r \ x$ is similar. It first checks whether $xl$ is odd (held in write mode or being promoted). If so then the implementation spins until $xl$ is even (free or held in read mode), at which point its value is incremented by two (to increase the number of readers by one) using the atomic ‘fetch-and-add’ (FAA) operation, and $x$ is successfully acquired in read mode. Dually, the implementation of $\text{unlock}_r \ x$ atomically decrements the value of $xl$ by two to decrease the reader count by one.

### Synchronisation Guarantees

In what follows we demonstrate that our implementation satisfies the ($\text{WSync}$), ($\text{WEx}$) and ($\text{RShare}$) conditions in Def. 4, as well as the stronger ($\text{RSync}$) condition discussed on page 16.

Observe that a successful acquisition of a writer lock is done via an atomic update operation (when the $\text{CAS}$ succeeds). That is, a call to $\text{lock}_w \ x$ returns only when the $\text{CAS}$ is successful, i.e. when no other thread holds a lock on $x$. Similarly, a call to $\text{unlock}_w \ x$ returns after an atomic write to $xl$ assigning it to zero. Moreover, once $x$ is acquired in write mode by a thread $\tau$, no other thread can acquire it (all other calls to $\text{lock}_r \ x$ and $\text{lock}_w \ x$ spin until $x$ is released by $\tau$). As such, any RA-consistent execution graph of a program $P$ containing a call by thread $\tau$ to $\text{lock}_w \ x$ followed by its subsequent release via $\text{unlock}_w \ x$ includes a trace of the following form, where $wl$ and $wu$ are events of thread $\tau$, $wl$ denotes the update event associated with the successful $\text{CAS}$ acquiring the writer lock, and $wu$ denotes the write event associated with its release:

$$
\frac{\rightarrow}{wl : U(xl, 0, 1) \stackrel{\text{po}_{\text{win}}}{\rightarrow} wu : W(xl, 0) \text{ (1)}}
$$

Note that if the use of locks in $P$ is well-formed (i.e. the sequence of lock events in each thread following $\text{po}$ matches (a prefix of) the regular expression $(\text{RL} \cdot \text{RU} \mid \text{WL} \cdot \text{WU} \mid \text{RL} \cdot \text{PL} \cdot \text{WU})^*$ – see page 15), no other write event on $xl$ can happen between $wl$ and $wu$. This is because i) well-formedness of lock traces ensures that the lock is released by the acquiring thread $\tau$ itself, and ii) the lock
on $x$ cannot be acquired after $wl$ and before $wu$: calls to $\text{lock}_v x$ fail because $x_1$ is non-zero; and calls to $\text{can-promote} x$ and $\text{lock}_r x$ fail because $x_1 = 1$ holds an odd value. This in turn ensures the mutual exclusion property of writer locks, as required by the (WEX) condition in Def. 4.

Similarly, a call to $\text{can-promote} x$ returns after the CAS succeeds and subsequently the condition of the while loop amounts to false and no other thread owns a lock on $x$ (in read or write mode). As such, any RA-consistent execution graph of a program $P$ containing a call by thread $\tau$ to $\text{can-promote} x$ followed by its subsequent release via $\text{unlock}_v x$ includes a trace of the following form, where $sp$, $pl$ and $wu$ are events of thread $\tau$, $sp$ denotes the update event associated with the successful CAS signalling promotion, $pl$ denotes the atomic read event in the final iteration of the while loop denoting successful promotion, and $wu$ denotes the write event associated with its release:

$$\quad \overset{rf}{\rightarrow} sp : U(x_1, 2i+2, 2i+1) \overset{rf}{\rightarrow} b_1 \overset{rf}{\rightarrow} \cdots \overset{rf}{\rightarrow} b_i \overset{rf}{\rightarrow} pl : R(x_1, 1) \overset{po_{x_1|m+1}}{\rightarrow} \overset{wu}{\rightarrow} W(x_1, 0) \overset{(2)}{\rightarrow}$$

for some $i$ denoting the reader count on $x$, with $b_1 \cdots b_n \in \mathcal{RU}_x$, $pl \overset{po_{x_1|m+1}}{\rightarrow} ru$. In other words, once $\tau$ has signalled its intention to promote $x$, no other lock on $x$ can be acquired – calls to $\text{lock}_v x$ fail as the value of $x_1$ is non-zero; calls to $\text{lock}_r x$ fail as $x_1$ holds an odd value. However, existing reader locks must be released before $x$ can be successfully promoted. As with writer locks above, if the use of locks in $P$ is well-formed, no other event on $x_1$ can happen between $pl$ and $wu$. This once again ensures the mutual exclusion property of promoted locks, as required by the (WEX) condition in Def. 4.

Analogously, a call to $\text{lock}_r x$ returns when the CAS succeeds; and a call to $\text{unlock}_r x$ returns after the atomic FAA operation. Moreover, once $x$ is acquired in reader mode by a thread $\tau$, no other thread can acquire it in write mode (all other calls to $\text{lock}_w x$ spin) until it has been released by $\tau$ (and potentially other readers). As such, any RA-consistent execution graph of a program $P$ containing a call to $\text{lock}_r x$ followed by its subsequent release via $\text{unlock}_r x$ includes a trace of the following form, where $rl$ and $ru$ are events of thread $\tau$, $rl$ denotes the update event associated with the successful CAS acquiring the reader lock, and $ru$ denotes the update event associated with its release via FAA:

$$\quad \overset{rf}{\rightarrow} rl : U(x_1, 2i, 2i+2) \overset{rf}{\rightarrow} a_1 \overset{rf}{\rightarrow} \cdots \overset{rf}{\rightarrow} a_n \overset{rf}{\rightarrow} sp : U(x_1, 2k+2, 2k+1) \overset{rf}{\rightarrow} b_1 \overset{rf}{\rightarrow} \cdots \overset{rf}{\rightarrow} b_m \overset{ru}{\rightarrow}$$

$$\quad \overset{rf}{\rightarrow} ru : U(x_1, j+2, j) \overset{(3)}{\rightarrow}$$

for some $i, j, k > 0$, where $a_1 \cdots a_n \in \mathcal{RL}_x \cup \mathcal{RU}_x$, $b_1 \cdots b_m \in \mathcal{RU}_x$, $rl \overset{po_{x_1|m}}{\rightarrow} ru$. In other words, other threads may also acquire $x$ in read mode (denoted by events of $a_1 \cdots a_i$) and their calls may be interleaved between the read lock and unlock of $\tau$. Moreover, another thread may signal to promote its reader lock on $x$ (denoted by $sp$) in between the acquisition and release of the reader lock on $x$ by $\tau$. Lastly, once the lock has been signalled for promotion, no other reader lock on $x$ can be acquired, though existing reader locks may be released (denoted by
\(b_1 \cdots b_m\)). Note that if the use of locks in \(P\) is well-formed, no thread can acquire \(x\) in write mode (as \(x\) holds a non-zero value) or successfully promote it before \(x\) is released by \(\tau\) via \(ru\). More concretely, threads may signal their intention to promote (see \(sp\) above). However, they cannot successfully promote it before \(\tau\) has released it in \(ru\) as \(x1\) holds a value greater than 1 (see \(pl\) in (2)). This ensures the (RSHARE) condition in Def. 4.

Observe that given an RA-consistent execution graph involving calls to the above MRSW lock library above, for each location \(x\) and its lock at \(x1\), the trace of events on \(x1\) comprises \(po\) and \(rf\) edges, as demonstrated by the traces in (1), (2) and (3). In other words, any two events on \(x1\) are related by \((po \cup rf)^+_loc\), i.e. \((po \cup rf)^+_loc\) is total for each \(x1\). Moreover, from the RA-consistency of our execution we know that \((po \cup rf)^+_loc\) is acyclic. As such, since \((po \cup rf)^+_loc\) is transitively closed, we know that it is a strict total order. For each location \(x\), we thus define \(lo_x\) as the strict total order given by \(lo_x \triangleq (po_{x1} \cup rf_{x1})^+\). As such, our implementation satisfies both the (WSync) condition in Def. 4 and the stronger (RSync) condition discussed on page 16.

### A.2 Write Synchronising MRSW Lock Implementation

Our second MRSW lock implementation is similarly implemented in the RA fragment of C11 [23] and is given in Fig. 8. In this implementation, each lock \(x\) is represented as an ordered map at location \(x+1\), written \(x1\). The map at \(x1\) contains one entry per thread as follows. For each thread with identifier \(\tau\), the \(x1[\tau]\) map entry records the current locking privileges of \(\tau\) on \(x\). More concretely, when \(x1[\tau] = 0\), then \(\tau\) does not hold the \(x\) lock; when \(x1[\tau] = 2\), then \(\tau\) holds \(x\) in read mode; and when \(x1[\tau] = 1\); then some thread (either \(\tau\) or another thread) either holds \(x\) in write mode, or it is in the process of acquiring \(x\) in write mode. The \(x\) lock is held in write mode only when all entries in \(x1\) are mapped to one. As we describe shortly, for thread \(\tau\) to acquire \(x\) in write mode, it must inspect each entry in \(x1\) (in order), wait for it be free (zero) and then set it to one. As we discuss shortly, this in-order inspection of entries allows us to avoid deadlocks. In our implementation, we assume that the thread identifier can be obtained by calling \(getTID\). We identify the top-most thread by \(\tau = 0\); as such, the entry of top-most thread in each map is ordered before all other threads.

We proceed with a more detailed explanation of our implementation after introducing our map notation. We write \(1\) to denote a map where all entries have value 1; similarly, we write \(0\) to denote a map where all entries have value 0. Lastly, we write \(S \subseteq x1\), to denote that the values held in map \(x1\) are a superset of \(S\). The lock map \(x1\) associated with location \(x\) can be in one of the following states:

- \(x1 = 0\) when \(x\) is free;
- \(x1 = 1\) when \(x\) is held in write mode;
- \(\{2\} \subseteq x1\) when \(x\) is held in read mode (by those threads \(\tau\) where \(x1[\tau] = 2\)).

When thread \(\tau\) calls \texttt{lock}_\(r\) \(x\), it simply spins until the lock is free (\(x1 = 0\) and thus \(x1[\tau] = 0\)), at which point it acquires it in read mode by setting \(x1[\tau]\) to two. Dually, when \(\tau\) calls \texttt{unlock}_\(r\) \(x\) it simply sets \(x1[\tau]\) to zero.
Fig. 8: Write synchronising MRSW lock implementation in the RA fragment of C11

Analogously, when \( \tau \) calls \text{lock}_w \ x, \) it traverses the \( xL \) map in order, spinning on each entry until it is free (0) and subsequently acquiring it (by setting it to 1). Conversely, when \( \tau \) calls \text{unlock}_w \ x, \) it releases \( x \) by traversing \( xL \) in order and setting each entry to one.

To understand the implementation of lock promotion, first consider the case where \text{can-promote} \ x \) is called by \( \tau \neq 0 \), i.e. a thread other than the top-most thread. The implementation of \text{can-promote} \ x \) then inspects the first entry in the map \( (xL[0]) \), i.e. that of the top-most thread. If \( xL[0] = 1 \), then \( x \) is currently being acquired by another thread; the promotion thus fails and false is returned. If on the other hand \( xL[0] \neq 1 \) (i.e. \( xL[0] = 0 \) or \( xL[0] = 2 \)), the implementation spins until it is zero and atomically updates it to one, signalling its intention to promote \( x \). This pre-empts the promotion of \( x \) by other threads: any such attempt would fail as now \( xL[0] = 1 \). The implementation then sets its own entry \( (xL[\tau]) \) to one, traverses the map in order, and spins on each entry until they too can be set to one. At this point the lock is successfully promoted and true is returned. Note that it is safe for \( \tau \) to update its own entry \( xL[\tau] \) to one: at this point in execution no thread holds the writer lock on \( x \), no thread can promote its lock on \( x \), and those threads with a reader lock on \( x \) never access the \( xL[\tau] \) entry – the read lock calls of another thread \( \tau' \) solely accesses \( xL[\tau'] \).

Let us now consider the case when the top-most thread with \( \tau = 0 \) calls \text{can-promote} \ x. \) Since prior to a \text{can-promote} \ x \) call \( \tau \) owns a reader lock on \( x \), i.e. \( xL[\tau] = 2 \), no other thread can promote its \( x \) lock. As such, \( \tau \) successfully sets \( xL[\tau] \) to one, signalling its intention to promote \( x \). In other words, the promotion is skewed in favour of the top-most thread: if a thread races against the top-most thread to promote \( x \), the top-most thread always wins. With the exception of the top-most thread, promotion is done on a ‘first-come-first-served’
basis. The rest of the implementation is then carried out as before: the map \( x1 \) is traversed in turn and each entry is set to one.

**Synchronisation Guarantees** In what follows we demonstrate that our implementation satisfies the (\( \text{WSync} \)), (\( \text{WEx} \)) and (\( \text{RShare} \)) conditions in Def. 4, while it does not satisfy the stronger (\( \text{RSync} \)) condition discussed on page 16.

Observe that a successful acquisition of a writer lock is done via several atomic update operations (via successful \( \text{CAS} \) operations in the for loop). That is, a call to \text{lock}_w x returns only once the \( \text{CAS} \) on all \( x1 \) entries succeeds, i.e. when no other thread holds a lock on \( x \). Similarly, a call to \text{unlock}_w x returns after atomic writes on each entry in \( x1 \), assigning them to zero. Moreover, once \( x \) is acquired in write mode by a thread \( \tau \), no other thread can acquire it: all other calls to \text{lock}_r x spin until the relevant \( x1 \) entry is set to zero by \( \tau \); all calls to \text{lock}_w x spin until all entries in \( x1 \) are set to zero by \( \tau \). As such, any RA-consistent execution graph of a program \( P \) containing a call by thread \( \tau \) to \text{lock}_w x or \text{can-promote} x, followed by its subsequent release via \text{unlock}_w x includes a trace of the following form, where the domain of the \( x1 \) map is \( 0 \cdots n \) for some \( n \), the \( wl_0, \cdots, wl_n, wu_0, \cdots, wu_n \) are events of thread \( \tau \), each \( wl_i \) denotes the update event associated with the successful \( \text{CAS} \) (in \text{lock}_w x) on the \( x1[i] \) entry, and each \( wu_i \) denotes the write event setting the \( x1[i] \) entry to zero (in \text{unlock}_w x):

\[
rf \rightarrow wl_0 : \text{U}(x1[0], 0, 1) \\
\downarrow \text{po} \\
\vdots \\
\downarrow \text{po} \\
rf \rightarrow wl_n : \text{U}(x1[n], 0, 1) \xrightarrow{\text{po}} wu_0 : \text{W}(x1[0], 0) \xrightarrow{\text{po}} \cdots \xrightarrow{\text{po}} wu_n : \text{W}(x1[0], 0)
\]  
(4)

As before, note that if the use of locks in \( P \) is well-formed (see page 15), no other write event on \( x1[i] \) can happen between \( wl_i \) and \( wu_i \). This is because no thread can acquire the write lock on \( x1 \) after \( wl_0 \) and before \( wu_n \); and thread \( \tau_i \) cannot acquire the read lock on \( x1 \) after \( wl_i \) and before \( wu_i \). This in turn ensures the mutual exclusion property of writer locks, as required by the (\( \text{WEx} \)) condition in Def. 4.

Analogously, a call to \text{lock}_r x returns when the \( \text{CAS} \) succeeds; and a call to \text{unlock}_r x returns after the atomic write operation setting \( x1[\tau] \) to zero. As such, any RA-consistent execution graph of a program \( P \) containing a call to \text{lock}_r x by thread \( \tau \), followed by its subsequent release via \text{unlock}_r x includes a trace of the following form, where \( rl \) and \( ru \) are events of thread \( \tau \), \( rl \) denotes the update event associated with the successful \( \text{CAS} \) acquiring the reader lock, and \( ru \) denotes the write event associated with its release by assigning it to zero:

\[
rf \rightarrow rl : \text{U}(x1[\tau], 0, 2) \xrightarrow{\text{po}} ru : \text{W}(x1[\tau], 0)
\]  
(5)

Note that each read lock call on \( x \) by thread \( \tau \), accesses \( x1[\tau] \) alone and no other entry in \( x1 \). As such, two threads may simultaneously acquire the reader
lock on x, ensuring the (RShare) condition in Def. 4. Moreover, as read lock calls by two distinct threads τ, τ′ access disjoint memory locations (x1[τ] and x1[τ′]), they never synchronise. That is, the MRSW lock implementation in Fig. 8 does not satisfy the (WSync) axiom on page 16.

Lastly, we demonstrate that our implementation satisfies the (WSync) condition in Def. 4, when accessed by an arbitrary (finite) number of threads n. Observe that given an RA-consistent execution graph involving calls to the MRSW lock library above, for each location x, its lock at x, and each thread τ, the trace of events on x1[τ] comprises po and rf edges, as demonstrated by the traces in (4) and (5). In other words, any two event on x1[τ] are related by (po ∪ rf)loc, i.e. (po ∪ rf)loc is total for each x1[τ]. We thus use this total order to determine the lo (synchronisation order) between two lock events, where at least one of them is a write lock event. Let us now pick two distinct lock events, w, l ∈ Lx, where at least one of them w is a write lock event, i.e. w ∈ 𝒮[Lx]. Either 1) w, l are events of the same thread; or 2) w and l are events of distinct threads τ, τ′, respectively. In the first case the two events are related by po one way or another, and as po ⊆ hb, we know that the two events synchronise. In the second case, there are two additional cases to consider: either i) l ∈ 𝒮[Lx], po and rf edges, as demonstrated by the traces in (4) and (5). In other words, any two event on x1[τ] are related by (po ∪ rf)loc, i.e. (po ∪ rf)loc is total for each x1[τ]. We thus use this total order to determine the lo (synchronisation order) between two lock events, where at least one of them is a write lock event. Let us now pick two distinct lock events, w, l ∈ Lx, where at least one of them w is a write lock event, i.e. w ∈ 𝒮[Lx]. Either 1) w, l are events of the same thread; or 2) w and l are events of distinct threads τ, τ′, respectively. In the first case the two events are related by po one way or another, and as po ⊆ hb, we know that the two events synchronise. In the second case, there are two additional cases to consider: either i) l ∈ 𝒮[Lx], po and rf edges, as demonstrated by the traces in (4) and (5). In other words, any two event on x1[τ] are related by (po ∪ rf)loc, i.e. (po ∪ rf)loc is total for each x1[τ]. We thus use this total order to determine the lo (synchronisation order) between two lock events, where at least one of them is a write lock event. Let us now pick two distinct lock events, w, l ∈ Lx, where at least one of them w is a write lock event, i.e. w ∈ 𝒮[Lx]. Either 1) w, l are events of the same thread; or 2) w and l are events of distinct threads τ, τ′, respectively. In the first case the two events are related by po one way or another, and as po ⊆ hb, we know that the two events synchronise. In the second case, there are two additional cases to consider: either i) l ∈ 𝒮[Lx], po and rf edges, as demonstrated by the traces in (4) and (5). In other words, any two event on x1[τ] are related by (po ∪ rf)loc, i.e. (po ∪ rf)loc is total for each x1[τ]. We thus use this total order to determine the lo (synchronisation order) between two lock events, where at least one of them is a write lock event. Let us now pick two distinct lock events, w, l ∈ Lx, where at least one of them w is a write lock event, i.e. w ∈ 𝒮[Lx]. Either 1) w, l are events of the same thread; or 2) w and l are events of distinct threads τ, τ′, respectively. In the first case the two events are related by po one way or another, and as po ⊆ hb, we know that the two events synchronise. In the second case, there are two additional cases to consider: either i) l ∈ 𝒮[Lx], po and rf edges, as demonstrated by the traces in (4) and (5). In other words, any two event on x1[τ] are related by (po ∪ rf)loc, i.e. (po ∪ rf)loc is total for each x1[τ].
B Auxiliary Results

Proposition 2 (SI-consistency). An execution graph $G$ is SI-consistent if and only if INT holds and the ‘SI-happens-before’ relation $si$-hb $\triangleq (po_T \cup rf_T \cup mo_T \cup si$-rb$)^+$ is irreflexive, where $si$-rb $\triangleq [R_E];rb_T;[W]$ and $R_E \triangleq \text{codom}(rf_E)$.

Proof. Pick an arbitrary execution graph $G$. We are then required to show:

\[
\text{acyclic}((po_T \cup rf_T \cup mo_T);rb_T^2) \iff \text{irreflexive}(si$-hb$)
\]

The $\Rightarrow$ direction
We proceed by contradiction. Assume that $\text{acyclic}((po_T \cup rf_T \cup mo_T);rb_T^2)$ and $\neg\text{irreflexive}(si$-hb$)$ both hold. We then know there exists $e$ such that $(e,e) \in si$-hb $= (po_T \cup rf_T \cup mo_T \cup si$-rb$)^+$. Note that $si$-rb; $si$-rb $= \emptyset$ because $R_E \cap W = \emptyset$. Hence, the $si$-hb cycle cannot have adjacent $si$-rb edges, which means that we have a cycle in $((po_T \cup rf_T \cup mo_T);si$-rb$^2)^+$, which contradicts our first assumption as $si$-rb $\subseteq rb_T$.

The $\Leftarrow$ direction
We proceed by contradiction. Let us assume $\text{irreflexive}(si$-hb$)$ and $\neg\text{acyclic}((po_T \cup rf_T \cup mo_T);rb_T^2)$. We then know there exists $e$ such that $(e,e) \in ((po_T \cup rf_T \cup mo_T);rb_T^2)^+$, i.e. there exists $e$ such that $(e,e) \in (rb_T^2;(po_T \cup rf_T \cup mo_T))^+$. There are now two cases to consider: either 1) $(e,e) \in (po_T \cup rf_T \cup mo_T)^+$; or 2) $\exists a,d.(e,a) \in (rb_T^2;(po_T \cup rf_T \cup mo_T))^*$, $(a,d) \in rb_T$; $po_T \cup rf_T \cup mo_T)$ and $(d,e) \in (rb_T^2;(po_T \cup rf_T \cup mo_T))^*$, i.e. $\exists a,(a,a) \in rb_T$; $po_T \cup rf_T \cup mo_T)$.

In case (1) from the definition of $si$-hb we then have $(e,e) \in si$-hb, contradicting the assumption that $\text{irreflexive}(si$-hb$)$ holds.

In case (2) we then know there exists $b,c$ such that $[b]_{st} \neq [c]_{st}$, $(a,b) \in rb_T$, $(b,c) \in po_T \cup rf_T \cup mo_T$, and $(a,c) \in rb_T^2;(po_T \cup rf_T \cup mo_T)^*$, i.e. $[b]_{st} \times [c]_{st} \subseteq po_T \cup rf_T \cup mo_T$. And from the definition of $si$-hb we have $[b]_{st} \times [c]_{st} \subseteq si$-hb. As such, from Lemma 1 below we have $[b]_{st} \times [a]_{st} \subseteq si$-hb. On the other hand, from $(a,b) \in rb_T$ we know there exist $r,w,w$ such that $r \in [a]_{st}$, $w \in [b]_{st}$, $[a]_{st} \neq [b]_{st}$, $(r,w) \in rb_T$, $(w',w) \in rf_T$ and $(w',w) \in mo_T$. There are now two cases to consider: i) $w' \in [a]_{st}$; or ii) $w' \notin [a]_{st}$.

In case (2.i) we then have $[a]_{st} \times [b]_{st} \subseteq mo_T \subseteq si$-hb. As such, we have $(a,b) \in si$-hb. Since we have also established $[b]_{st} \times [a]_{st} \subseteq si$-hb, we have $(a,b) \in si$-hb. By transitivity, we have $(a,a) \in si$-hb, contradicting the assumption that $\text{irreflexive}(si$-hb$)$ holds.

In case (2.ii) we then know $r \in R_E$. As such we have $r,w) \in si$-rb $\subseteq si$-hb. Since we have also established $[b]_{st} \times [a]_{st} \subseteq si$-hb, we have $(w,r) \in si$-hb. By transitivity, we have $(r,r) \in si$-hb, contradicting the assumption that $\text{irreflexive}(si$-hb$)$ holds.
**Theorem 5.** For all executions $G$, if $G \cdot \mathcal{N} = \emptyset$, then:

\[ G \text{ is SI-consistent } \iff G \text{ is RSI-consistent} \]

*Proof (the $\Leftarrow$ direction).* Pick an arbitrary $G$ such that $G \cdot \mathcal{N} = \emptyset$ and $G$ is RSI-consistent. Let us proceed by contradiction and assume $G$ is not SI-consistent. That is, there exists a such that $(a, a) \in G \cdot \text{si-hb}$. From the definition of si-hb we then have $G \cdot \text{si-hb} \subseteq G \cdot \text{rsi-hb}$. As such, we have $(a, a) \in G \cdot \text{si-hb}$, contradicting the assumption that $G$ is RSI-consistent.

*Proof (the $\Rightarrow$ direction).* Pick an arbitrary $G$ such that $G \cdot \mathcal{N} = \emptyset$ and $G$ is SI-consistent. As such, since from (INT) we have $G \cdot \text{mo} \subseteq G \cdot \text{mo}_T \cup G \cdot \text{po}_I$. Moreover, we have $G \cdot \text{mo}_I \subseteq G \cdot \text{mo}_T \cup G \cdot \text{po}_I$. From the definition of si-hb and since $G \cdot \text{mo}_I$ is irreflexive, we have $G \cdot \text{mo}_I \subseteq A$ and thus $G \cdot \text{mo} \subseteq G \cdot \text{rsi-hb}_\text{loc}$.

Let $B \equiv G \cdot \text{rb} \cap G \cdot \text{po}_I$. We also have $G \cdot \text{rb} \subseteq G \cdot \text{rsi-hb} \cup G \cdot \text{rb}_I$. As such, since from (INT) we have $G \cdot \text{rb}_I \subseteq G \cdot \text{po}_I$, we have $G \cdot \text{rb}_I \subseteq B$ and thus $G \cdot \text{rb} \subseteq G \cdot \text{rsi-hb}_\text{loc} \cup B$. Consequently, we have $G \cdot (\text{rsi-hb}_\text{loc} \cup \text{mo} \cup \text{rb}) = G \cdot (\text{rsi-hb}_\text{loc} \cup B) \subseteq G \cdot (\text{rsi-hb}_\text{loc} \cup A_{\text{loc}} \cup B)$. Therefore, $G \cdot (\text{rsi-hb}_\text{loc} \cup \text{mo} \cup \text{rb})^+ = G \cdot (\text{si-hb}_\text{loc} \cup A_{\text{loc}} \cup B)^+$. Observe that $A; B = \emptyset$. In what follows we demonstrate that:

\[ A; \text{si-hb} \subseteq \text{si-hb} \quad \text{and} \quad \text{si-hb}; A \subseteq \text{si-hb} \]  \hspace{1cm} (6)

As such, since $\text{si-hb}$ is transitively closed, we have $G \cdot \text{rsi-hb} = G \cdot \text{si-hb} \cup A$. Moreover, we have $G \cdot \text{mo} \subseteq G \cdot \text{mo}_T \cup G \cdot \text{mo}_I$. As such, from (INT) we have $G \cdot \text{mo}_I \subseteq G \cdot \text{po}_I$, we have $G \cdot \text{mo}_I \subseteq A$ and thus $G \cdot \text{mo} \subseteq G \cdot \text{rsi-hb}_\text{loc}$.

Note that since $G$ is SI-consistent we know that $A$ is irreflexive. As such, to show that $G \cdot (\text{rsi-hb}_\text{loc} \cup \text{mo} \cup \text{rb})^+ = G \cdot (\text{si-hb}_\text{loc} \cup A_{\text{loc}} \cup B)^+$ is irreflexive, from (6), (7), (8), and since $B$ is irreflexive, it suffices to show that $G \cdot \text{si-hb}$ is irreflexive, which follows immediately from the SI-consistency of $G$.

**TS.** (6)

To show $A; \text{si-hb} \subseteq \text{si-hb}$, pick an arbitrary $(a, b) \in A; \text{si-hb}$. We then know there exists $c$ such that $a, c \in W$, $(a, c) \in A$ and $(c, b) \in \text{si-hb}$. From the definition of si-hb and since $c \in W$, we know there exists $d$ such that $(c, d) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T$ and $(d, b) \in \text{si-hb}^*$. As $(a, c) \in A \subseteq \text{st}$, we thus know that $(a, d) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \subseteq \text{si-hb}$. Consequently, since $(d, b) \in \text{si-hb}^*$ and si-hb is transitively closed, we have $(a, b) \in \text{si-hb}$, as required.

To show $\text{si-hb}; A \subseteq \text{si-hb}$, pick an arbitrary $(a, b) \in \text{si-hb}; A$. We then know there exists $c$ such that $b, c \in W$, $(a, c) \in \text{si-hb}$ and $(c, b) \in A$. From the definition of si-hb, we know there exists $d$ such that $(a, d) \in \text{si-hb}^*$ and $(d, c) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \subseteq \text{si-hb}$. As $(c, b) \in A \subseteq \text{st}$ and since $b, c \in W$ we thus know that $(d, b) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \cup \text{si-rb} \subseteq \text{si-hb}$. Consequently, since $(a, d) \in \text{si-hb}^*$ and si-hb is transitively closed, we have $(a, b) \in \text{si-hb}$, as required.
**TS. (7)**

Pick an arbitrary \( a, b \) such that \((a, b) \in B; A \). We then know there exists \( c \) such that \( b, c \in W, (a, c) \in B, (c, b) \in A \) and \( \text{tx}(a) = \text{tx}(b) = \text{tx}(c) \). As \( b, c \in W \), we know either \((b, c) \in \text{mo}_1 \) or \((c, b) \in \text{mo}_1 \). However, as \( G \) is SI-consistent, from (INT) we know that \( \text{mo}_1 \subseteq \text{po}_1 \) and thus we have \((c, b) \in \text{mo}_1 \cap \text{po}_1 \). As such, since \((a, c) \in \text{rb} \cap \text{po}_1 \), \((c, b) \in \text{mo}_1 \cap \text{po}_1 \), and \( \text{po}_1 \) is transitively closed, from the definition of \( \text{rb} \) we have \((a, b) \in \text{rb} \cap \text{po}_1 = B \), as required.

**TS. (8)**

To show \( B; \text{si-hb} \subseteq \text{si-hb}; B \), pick an arbitrary \((a, b) \in B; \text{si-hb} \). We then know there exists \( c \) such that \( a \in R, c \in W, (a, c) \in B \) and \((c, b) \in \text{si-hb} \). From the definition of \( \text{si-hb} \) and since \( c \in W \), we know there exists \( d \) such that \((c, d) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \) and \((d, b) \in \text{si-hb}^* \). As \((a, c) \in B \subseteq \text{st} \), we thus know that \((a, d) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \subseteq \text{si-hb} \). Consequently, since \((d, b) \in \text{si-hb}^* \) and \( \text{si-hb} \) is transitively closed, we have \((a, b) \in \text{si-hb} \), as required.

To show \( \text{si-hb}; B \subseteq \text{si-hb}; B \), pick an arbitrary \((a, b) \in \text{si-hb}; B \). We then know there exists \( c \) such that \( c \in R, b \in W, (a, c) \in \text{si-hb} \) and \((c, b) \in B \). From the definition of \( \text{si-hb} \), and since \( c \in R \), we know there exists \( d \) such that \((a, d) \in \text{si-hb}^* \) and \((d, c) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \). As \((c, b) \in B \subseteq \text{st} \), we thus know that \((d, b) \in \text{po}_T \cup \text{rf}_T \cup \text{mo}_T \subseteq \text{si-hb} \). Consequently, since \((a, d) \in \text{si-hb}^* \) and \( \text{si-hb} \) is transitively closed, we have \((a, b) \in \text{si-hb} \), as required.
Theorem 6 (Monotonicity). Let $G_T$ be an RSI execution graph obtained from an RSI execution graph $G$ by wrapping some non-transactional events inside a new transaction. If $G_T$ is RSI-consistent, then so is $G$.

Proof. First, we show that $(W); G_.rsi-hb \subseteq G_T.po_1 \cup G_T.rsi-hb$. Let $A$ denote the events of the new transactions in $G_T$. Let $(a, b) \in [W]; G_.rsi-hb$, and consider a path from $a$ to $b$ of minimal length (all relations are in $G$). Note that the only possible edges on this path that do not appear in a corresponding relation in $G_T$ are $[A]; rf; [A]$ and $[A]; po; [A] \setminus (W \times W)$ edges. Suppose first that $[A]; rf; [A]$ is used on this path. Immediately before or after $[A]; rf; [A]$, we can only have $po \setminus po_1$, but then the two edges can be replaced by a shorter path that uses only $po \setminus po_1$. Thus, in this case it follows that $(a, b) \in [A]; rf; [A] \subseteq G_T.po_1$. Next, suppose that $[A]; po; [A] \setminus (W \times W)$ is used on this path (the minimality ensures such edge is used only once). First, if this edge is the first edge on the path, then (again) it is the only edge on the path (immediately after $[A]; po; [A]$, we can only have $rf; [N'T \setminus A]$ or $[N'T]; rf; st$, but both start with a write). Hence, in this case we have $(a, b) \in [A]; po; [A] \subseteq G_T.po_1$. Second, consider the case that the $[A]; po; [A]$ edge is not the first on the path. Immediately before $[A]; po; [A]$, we can only have a $[E \setminus A]; rf$ edge. Then, in $G_T$, the two edges can be joined into either $[N'T]; G_T.rf; G_T.st$ or $G_T.rf_T$. Hence, we obtain $(a, b) \in G_T_.rsi-hb$.

Now, suppose that $G$ is not RSI-consistent. If INT does not hold, then $G_T$ is also not RSI-consistent and we are done. Otherwise, $G_.rsi-hb.loc \cup G_.mo \cup G_.rb$ is cyclic. Since $G_.mo$ is total on writes to each location, it follows that $(a, a) \in G_.rsi-hb; (G_.mo \cup G_.rb)$ for some $a \in G.W$. Since $G_.mo = G_T_.mo$, $G_.rb = G_T_.rb$, our claim above entails that $(a, a) \in (G_T.po_1 \cup G_T_.rsi-hb); (G_T_.mo \cup G_T_.rb)$. Now, $(a, a) \in G_T_.po_1; (G_T_.mo \cup G_T_.rb)$ implies that INT does not hold for $G_T$, while $(a, a) \in G_T_.rsi-hb; (G_T_.mo \cup G_T_.rb)$ implies that $G_T_.rsi-hb.loc \cup G_T_.mo \cup G_T_.rb$ is cyclic. In both cases, $G_T$ is not RSI-consistent.
Lemma 1. For all $a, b, c$:

$$[a]_{st} \neq [b]_{st} \land [a]_{st} \times [b]_{st} \subseteq \text{si-hb} \land (b, c) \in (rb_T); (po_T \cup rf_T \cup mo_T))^* \Rightarrow [a]_{st} \times [c]_{st} \subseteq \text{si-hb}$$

Proof. Note that $(rb_T); (po_T \cup rf_T \cup mo_T))^* = \bigcup_{n \in \mathbb{N}} S_n$, where $S_0 \triangleq \text{id}$ and for all $n \in \mathbb{N}$ we have $S_{n+1} \triangleq (rb_T); (po_T \cup rf_T \cup mo_T); S_n$. We thus demonstrate instead:

$$\forall a, b, c. \forall n \in \mathbb{N}. [a]_{st} \neq [b]_{st} \land [a]_{st} \times [b]_{st} \subseteq \text{si-hb} \land (b, c) \in S_n \Rightarrow [a]_{st} \times [c]_{st} \subseteq \text{si-hb}$$

We proceed by induction on $n$.

Base case $n = 0$

Follows immediately from the assumptions of the lemma and the definition of $S_0$.

Inductive case $n = m + 1$

$$\forall a, b, c. \forall i \leq m.$$

$$[a]_{st} \neq [b]_{st} \land [a]_{st} \times [b]_{st} \subseteq \text{si-hb} \land (b, c) \in S_i \Rightarrow [a]_{st} \times [c]_{st} \subseteq \text{si-hb} \quad (\text{I.H.})$$

Pick arbitrary $a, b, c$ such that $[a]_{st} \neq [b]_{st}$, $[a]_{st} \times [b]_{st} \subseteq \text{si-hb}$, and $(b, c) \in S_n$. From the definition of $S_n$ we then know there exists $d$ such that $(b, d) \in rb_T; (po_T \cup rf_T \cup mo_T)$ and $(d, c) \in S_m$. There are now two cases to consider: 1) $(b, d) \in po_T \cup rf_T \cup mo_T$; or 2) $(b, d) \in rb_T; (po_T \cup rf_T \cup mo_T)$.

In case (1) from the definitions of $po_T$, $rf_T$ and $mo_T$ we have $[b]_{st} \times [d]_{st} \subseteq po_T \cup rf_T \cup mo_T$ and thus from the definition of $\text{si-hb}$ we have $[a]_{st} \times [c]_{st} \subseteq \text{si-hb}$. Since we also have $[a]_{st} \times [b]_{st} \subseteq \text{si-hb}$ and $\text{si-hb}$ is transitivity closed, we have $[a]_{st} \times [d]_{st} \subseteq \text{si-hb}$. As $(d, c) \in S_m$, from (I.H.) we then have $[a]_{st} \times [c]_{st} \subseteq \text{si-hb}$, as required.

In case (2) we then know there exists $e$ such that $(b, e) \in rb_T$ and $(e, d) \in po_T \cup rf_T \cup mo_T$. From the definitions of $po_T$, $rf_T$ and $mo_T$ we then have $[e]_{st} \times [d]_{st} \subseteq po_T \cup rf_T \cup mo_T$ and thus from the definition of $\text{si-hb}$ we have $[e]_{st} \times [d]_{st} \subseteq \text{si-hb}$. On the other hand, from $(b, e) \in rb_T$ we know there exist $r, w, w'$ such that $r \in [b]_{st}$, $w \in [e]_{st}$, $[b]_{st} \neq [e]_{st}$, $[d]_{st} \neq [e]_{st}$, $(r, w) \in rb$, $(w', r) \in rf$ and $(w', w) \in mo$.

There are now two cases to consider: i) $w' \in [b]_{st}$; or ii) $w' \notin [b]_{st}$.

In case (2.i) we then have $[b]_{st} \times [e]_{st} \subseteq \text{mo_T} \subseteq \text{si-hb}$. Since we also have $[a]_{st} \times [b]_{st} \subseteq \text{si-hb}$ and $[e]_{st} \times [d]_{st} \subseteq \text{si-hb}$, from the transitivity of $\text{si-hb}$ we have $[a]_{st} \times [d]_{st} \subseteq \text{si-hb}$. As $(d, c) \in S_m$, from (I.H.) we then have $[a]_{st} \times [c]_{st} \subseteq \text{si-hb}$, as required.

In case (2.ii) we then know $r \in R_E$. As such we have $(r, w) \in sr \subseteq \text{si-hb}$. As $[a]_{st} \times [b]_{st} \subseteq \text{si-hb}$ and $r \in [b]_{st}$ we then have $[a]_{st} \times \{w\} \subseteq \text{si-hb}$, Similarly, as $[e]_{st} \times [d]_{st} \subseteq \text{si-hb}$ and $w \in [e]_{st}$ we then have $\{w\} \times [d]_{st} \subseteq \text{si-hb}$. As such, from the transitivity of $\text{si-hb}$ we have $[a]_{st} \times [d]_{st} \subseteq \text{si-hb}$. As $(d, c) \in S_m$, from (I.H.) we then have $[a]_{st} \times [c]_{st} \subseteq \text{si-hb}$, as required.
We proceed by contradiction. Let $wu, wu' \in \mathcal{WU}_x$, $rl, rl' \in \mathcal{RL}_x$, and $ru \in \mathcal{RU}_x$:

\begin{align*}
(wl, wu), (wl', wu') \in po_{x|imm} & \Rightarrow (wu, wl') \in lo_x \lor (wu', wl) \in lo_x \quad \text{(WWSYNC)} \\
(wl, wu), (rl, pl, wu') \in po_{x|imm} & \Rightarrow (wu, rl) \in lo_x \lor (wu', wu) \in lo_x \\
(rl, pl, wu), (rl', pl', wu') \in po_{x|imm} & \Rightarrow (wu, rl') \in lo_x \lor (wu', rl) \in lo_x
\end{align*}

where given a relation $r$ we write $(a, b, c) \in r$ as a shorthand for $(a, b), (b, c) \in r$.

Proof (Proof (WWSYNC)). Pick an arbitrary RA-consistent execution graph $G = (E, po, rl, mo, lo)$ of the SI or RSI implementations in Fig. 3, and pick arbitrary $wl, wu' \in \mathcal{WL}_x$, $pl, pl' \in \mathcal{PL}_x$, $wu, wu' \in \mathcal{WU}_x$ and $rl, rl' \in \mathcal{RL}_x$. We are then required to show:

\begin{align*}
(wl, wu), (wl', wu') \in po_{x|imm} & \Rightarrow (wu, wl') \in lo_x \lor (wu', wl) \in lo_x \quad \text{(9)} \\
(wl, wu), (rl, pl, wu') \in po_{x|imm} & \Rightarrow (wu, rl) \in lo_x \lor (wu', wl) \in lo_x \quad \text{(10)} \\
(rl, pl, wu), (rl', pl', wu') \in po_{x|imm} & \Rightarrow (wu, rl') \in lo_x \lor (wu', rl) \in lo_x \quad \text{(11)}
\end{align*}

RTS. (9) We proceed by contradiction. Let $(wl, wu), (wl', wu') \in po_{x|imm}$ and $(wu, wu') \notin lo_x \land (wu', wu) \notin lo_x$. Since $lo_x$ is totally ordered w.r.t. write lock events, we then have $(wu', wu) \in lo_x$ and $(wu, wu') \in lo_x$.

From the mutual exclusion (WEX) afforded by lock events, and since $(wl', wu) \in lo_x$ we know there exists $wu'' \in \mathcal{WU}_x$ such that $wl' \overset{po}{\rightarrow} wu''$ and $wu'' \overset{lo}{\rightarrow} wu$. Moreover, since $wl' \overset{po_{x|imm}}{\rightarrow} wu'$ we have $wu' \overset{po}{\rightarrow} wu''$. Once again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $wu' \overset{lo}{\rightarrow} wu''$. We then have $wu' \overset{lo}{\leftarrow} wu'' \overset{lo}{\rightarrow} wu$ and thus since $lo$ is an order (i.e. is transitive), we have $wu' \overset{lo}{\rightarrow} wu$.

Following a similar argument symmetrically, we get $wu \overset{lo}{\rightarrow} wu'$. We then have $wu' \overset{lo}{\rightarrow} wu \overset{lo}{\rightarrow} wu'$, contradicting the assumption that $lo$ is a strict order.

RTS. (10) We proceed by contradiction. Let $(wl, wu), (rl, pl, wu') \in po_{x|imm}$ and $(wu, rl) \notin lo_x \land (wu', pl) \notin lo_x$. Since $lo_x$ is totally ordered w.r.t. write lock events, we then have $(rl, wu) \in lo_x$ and $(wl, wu') \in lo_x$.

From (RSHARE) and since $(rl, wu) \in lo_x$ we know there exists $l \in \mathcal{RU}_x \cup \mathcal{PL}_x$ such that $rl \overset{pl}{\rightarrow} l$ and $l \overset{lo}{\rightarrow} wu$. Moreover, since $rl \overset{po_{x|imm}}{\rightarrow} pl$ we have $pl \overset{lo}{\rightarrow} l$. Once again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $pl \overset{lo}{\rightarrow} l$. We then have $pl \overset{lo}{\rightarrow} l \overset{lo}{\rightarrow} wu$ and thus since $lo$ is an order (i.e. is transitive), we have $pl \overset{lo}{\rightarrow} wu$. Similarly, from the mutual exclusion (WEX) of
write locks and since $pl \xrightarrow{lo} wu$, we know there exists $wu'' \in WU_x$ such that $pl \xrightarrow{po} wu'$ and $wu'' \xrightarrow{lo} wu$. Moreover, since $pl \xrightarrow{po\mid imm} wu'$ we have $wu' \xrightarrow{po'} wu''$. Again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $wu' \xrightarrow{lo} wu''$. We then have $wu' \xrightarrow{lo} wu'' \xrightarrow{lo} wu$ and thus since $lo$ is an order (i.e. is transitive), we have $wu' \xrightarrow{lo} wu$.

Analogously, from (WEx) and since $(wl, wu') \in lo$, we know there exists $wu''' \in WU_x$ such that $wl \xrightarrow{po} wu'''$ and $wu''' \xrightarrow{lo} wu'$. Moreover, since $wl \xrightarrow{po\mid imm} wu'$ we have $wu \xrightarrow{po'} wu'''$. Once again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $wu \xrightarrow{lo} wu'''$. We then have $wu \xrightarrow{lo} wu''' \xrightarrow{lo} wu'$ and thus since $lo$ is an order (i.e. is transitive), we have $wu \xrightarrow{lo} wu'$.

We then have $wu' \xrightarrow{lo} wu \xrightarrow{lo} wu'$, contradicting the assumption that $lo$ is a strict order.

**RTS. (11)**

We proceed by contradiction. Let $(rl, pl, wu), (rl', pl', wu') \in po\mid imm$ and $(wu, rl') \not\in lo_x \land (wu', rl) \not\in lo_x$. Since $lo_x$ is totally ordered w.r.t. write lock events, we then have $(rl, wu') \in lo_x$ and $(rl', wu) \in lo_x$.

From (RShare) and since $(rl, wu') \in lo_x$ we know there exists $l \in RU_x \cup \mathcal{P}_x$ such that $rl \xrightarrow{po} l$ and $l \xrightarrow{lo} wu'$. Moreover, since $rl \xrightarrow{po\mid imm} pl$ we have $pl \xrightarrow{po'} l$. Once again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $pl \xrightarrow{lo} l$. We then have $pl \xrightarrow{lo} l \xrightarrow{lo} wu'$ and thus since $lo$ is an order (i.e. is transitive), we have $pl \xrightarrow{lo} wu'$. Similarly, from (WEx) and since $pl \xrightarrow{lo} wu'$, we know there exists $wu'' \in WU_x$ such that $pl \xrightarrow{po} wu''$ and $wu'' \xrightarrow{lo} wu'$. Moreover, since $pl \xrightarrow{po\mid imm} wu'$ we have $wu \xrightarrow{po'} wu''$. Again, since $lo$ is total w.r.t. write lock events and agrees with $po$ (otherwise we would have a cycle contradicting the assumption that $G$ is RA-consistent), we have $wu \xrightarrow{lo} wu''$. We then have $wu \xrightarrow{lo} wu'' \xrightarrow{lo} wu'$ and thus since $lo$ is an order (i.e. is transitive), we have $wu \xrightarrow{lo} wu'$.

Following a similar argument symmetrically, we get $wu' \xrightarrow{lo} wu$. We then have $wu' \xrightarrow{lo} wu \xrightarrow{lo} wu'$, contradicting the assumption that $lo$ is a strict order.

**Proof (Proof (RWSync)).** Pick an arbitrary RA-consistent execution graph $G = (E, po, rf, mo, lo)$ of the SI or RSI implementations in Fig. 3, and pick arbitrary $wl \in WL_x$, $pl \in PL_x$, $wu \in WU_x$, $rl, rl' \in RL_x$ and $ru \in RU_x$. We are then required to show:

\[(wl, wu), (rl, ru) \in po\mid imm \Rightarrow (wu, rl) \in lo_x \lor (ru, wl) \in lo_x \quad (12)\]
\[(rl, pl, wu), (rl', ru) \in po\mid imm \Rightarrow (wu, rl') \in lo_x \lor (ru, pl) \in lo_x \quad (13)\]
RTS. (12)
We proceed by contradiction. Let \((wl, wu), (rl, ru) \in \text{po}_x\) and \((wu, rl) \notin \text{lo}_x \land (ru, wl) \notin \text{lo}_x\). Since \text{lo}_x is totally ordered w.r.t. write lock events, we then have \((rl, wu) \in \text{lo}_x\) and \((wu, rl) \in \text{lo}_x\).

From (WEx) and since \((wl, ru) \in \text{lo}_x\) we know there exists \(wu' \in \text{WU}_x\) such that \(wl \xrightarrow{\text{po}} wu'\) and \(wu' \xrightarrow{\text{lo}} ru\). Moreover, since \text{lo} is totally ordered w.r.t. write lock events and agrees with \text{po} (otherwise we would have a cycle contradicting the assumption that \(G\) is RA-consistent), we have \(wu \xrightarrow{\text{lo}} wu'\). We then have \(wu \xrightarrow{\text{lo}} wu' \xrightarrow{\text{lo}} ru\) and thus since \text{lo} is an order (i.e. is transitive), we have \(wu \xrightarrow{\text{lo}} ru\).

Analogously, from (RShare) and since \((rl, wu) \in \text{lo}_x\) we know there exists \(l \in \text{RU}_x \cup \text{PL}_x\) such that \(rl \xrightarrow{\text{po}} l\) and \(l \xrightarrow{\text{lo}} wu\). Moreover, since \(rl \xrightarrow{\text{po}} x\) we have \(ru \xrightarrow{\text{po}} l\). We then have \(l \xrightarrow{\text{lo}} wu \xrightarrow{\text{lo}} ru \xrightarrow{\text{lo}} l\), contradicting the assumption that \(G\) is RA-consistent.

RTS. (13)
We proceed by contradiction. Let \((rl, pl, wu), (rl', ru) \in \text{po}_x\) and \((wu, rl') \notin \text{lo}_x \land (ru, pl) \notin \text{lo}_x\). Since \text{lo}_x is totally ordered w.r.t. write lock events, we then have \((rl', wu) \in \text{lo}_x\) and \((pl, ru) \in \text{lo}_x\).

From (WEx) and since \((pl, ru) \in \text{lo}_x\) we know there exists \(wu' \in \text{WU}_x\) such that \(pl \xrightarrow{\text{po}} wu'\) and \(wu' \xrightarrow{\text{lo}} ru\). Moreover, since \(pl \xrightarrow{\text{po}} x\) we have \(wu \xrightarrow{\text{po}} wu'\). Once again, since \text{lo} is totally ordered w.r.t. write lock events and agrees with \text{po} (otherwise we would have a cycle contradicting the assumption that \(G\) is RA-consistent), we have \(wu \xrightarrow{\text{lo}} wu'\). We then have \(wu \xrightarrow{\text{lo}} wu' \xrightarrow{\text{lo}} ru\) and thus since \text{lo} is an order (i.e. is transitive), we have \(wu \xrightarrow{\text{lo}} ru\).

Analogously, from (RShare) and since \((rl', wu) \in \text{lo}_x\) we know there exists \(l \in \text{RU}_x \cup \text{PL}_x\) such that \(rl \xrightarrow{\text{po}} l\) and \(l \xrightarrow{\text{lo}} wu\). Moreover, since \(rl \xrightarrow{\text{po}} x\) we have \(ru \xrightarrow{\text{po}} l\). We then have \(l \xrightarrow{\text{lo}} wu \xrightarrow{\text{lo}} ru \xrightarrow{\text{lo}} l\), contradicting the assumption that \(G\) is RA-consistent.
C Soundness and Completeness of the Eager SI Implementation

Notation Given an execution graph \( G' = (E, po, rf, mo, lo) \) we write \( T/st \) for the set of equivalence classes of \( T \) induced by \( st \); \([a]_{st}\) for the equivalence class that contains \( a \); and \( T_\xi \) for the equivalence class of transaction \( \xi \in TXId \): 

\[ T_\xi = \{ a | tx(a) = \xi \} \]

We write \text{si-consistent}(G') to denote that \( G' \) is SI-consistent; and write \text{RA-consistent}(G) to denote that \( G \) is RA-consistent.

Given an execution graph \( G \) of the SI implementation in Fig. 3, let us assign a transaction identifier to each transaction executed by the program; and given a transaction \( \xi \), let \( RS_\xi \) and \( WS_\xi \) denote its read and write sets, respectively. Observe that given a transaction \( \xi \) of the SI implementation in Fig. 3 with 

\[ RS_\xi = \{ x_1, \cdots, x_i \} \]

the trace of \( \xi \), written \( T_\xi \), is of the following form:

\[ T_\xi = FS^* po_{imm} \rightarrow Rs po_{imm} \rightarrow RUs po_{imm} PLs po_{imm} Ts po_{imm} Us \]

where \( FS^* \) denotes the sequence of events attempting but failing to obtain a snapshot, and

\begin{itemize}
  \item \( Rs \) denotes the sequence of events acquiring the reader locks (on all locations accessed) and capturing a snapshot of the read set, and is of the form 
  \[ rl_{x_1} po_{imm} \rightarrow \cdots \rightarrow rl_{x_i} po_{imm} S_{x_1} po_{imm} \rightarrow \cdots \rightarrow S_{x_i} \]
  where for all \( n \in \{1\cdots i\} \):
  \[ rl_{x_n} = RL(x_1, n) \]
  \[ S_{x_n} = \begin{cases} rs_{x_n} po_{imm} \rightarrow ws_{x_n} & \text{if } x_n \in RS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]
  with \( rs_{x_n} \triangleq R(x_n, v_n) \) and \( ws_{x_n} \triangleq W(s[x_n], v_n) \), for some \( v_n \).
  \item \( RUs \) denotes the sequence of events releasing the reader locks (when the given location is in the read set only) and is of the form 
  \[ ru_{x_1} po_{imm} \rightarrow \cdots \rightarrow ru_{x_i} \]
  where for all \( n \in \{1\cdots i\} \):
  \[ ru_{x_n} = \begin{cases} RU(x_1, n) & \text{if } x_n \not\in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]
  \item \( PLs \) denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form 
  \[ pl_{x_1} po_{imm} \rightarrow \cdots pl_{x_i} \]
  where for all \( n \in \{1\cdots i\} \):
  \[ pl_{x_n} = \begin{cases} PL(x_1, n) & \text{if } x_n \in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]
\end{itemize}
– $Ts$ denotes the sequence of events corresponding to the execution of $(T)$ in Fig. 3 and is of the form $t_1^{po|_{imm}} \rightarrow \cdots \rightarrow t_k$, where for all $m \in \{1 \cdots k\}$:

$$t_m = \begin{cases} 
R(s[x_n], v_n) & \text{if } O_m = R(x_n, v_n) \\
\bar{w}(x_n, v_n) \xrightarrow{po|_{imm}} \bar{w}(s[x_n], v_n) & \text{if } O_m = \bar{w}(x_n, v_n)
\end{cases}$$

where $O_m$ denotes the $m$th event in the trace of the original $T$:

– $Us$ denotes the sequence of events releasing the locks on the write set. That is, the events in $Us$ correspond to the execution of the last line of the implementation in Fig. 3, and is of the form $wu_{x_1}^{po|_{imm}} \rightarrow \cdots \rightarrow wu_{x_i}$, where for all $n \in \{1 \cdots i\}$:

$$wu_{x_n} = \begin{cases} 
\bar{w}(y1_n) & \text{if } x_n \in \bar{W}s \\
\emptyset & \text{otherwise}
\end{cases}$$

Given a transaction trace $\theta_\xi$, we write e.g. $\xi.Ls$ to refer to its constituent $Ls$ sub-trace and write $Ls.E$ for the set of events related by po in $Ls$. Similarly, we write $\xi.E$ for the set of events related by po in $\theta_\xi$. Note that $G.E = \bigcup_{\xi \in Tx} \xi.E$.

### C.1 Implementation Soundness

In order to establish the soundness of our implementation, it suffices to show that given an RA-consistent execution graph of the implementation $G$, we can construct a corresponding SI-consistent execution graph $G'$ with the same outcome.

Given a transaction $\xi \in Tx$ with $RS_\xi \cup WS_\xi = \{x_1 \cdots x_i\}$ and trace $\theta_\xi = F_{S^* \xrightarrow{po|_{imm}} R_{S^* \xrightarrow{po|_{imm}} R_{Us} \xrightarrow{po|_{imm}} P_{L_{S^* \xrightarrow{po|_{imm}} T_{S^* \xrightarrow{po|_{imm}} Us}}}}) \rightarrow t_k$, we construct the corresponding implementation trace $\theta'_\xi$ as follows:

$$\theta'_\xi \triangleq t'_1 \xrightarrow{po} \cdots \xrightarrow{po} t'_k$$

where for all $m \in \{1 \cdots k\}$:

$$t'_m = R(x_n, rb_n) \text{ when } t_m = R(s[x_n], rb_n)$$

$$t'_m = \bar{w}(x_n, rb_n) \text{ when } t_m = \bar{w}(s[x_n], rb_n)$$

such that in the first case the identifier of $t'_m$ is that of $t_m$; and in the second case the identifier of $t'_m$ is that of the first event in $t_m$. We then define:

$$RF_\xi \triangleq \begin{cases} 
\{ t'_j \in Ts' \land \exists x,v. t'_j = R(x,v) \land w = \bar{w}(x,v) & \text{such that in the first case the identifier of } t'_m \text{ that of } t_m \text{; and in the second case the identifier of } t'_m \text{ that of the first event in } t_m. \text{ We then define:} \\
w \in \xi.E \Rightarrow w \xrightarrow{po} t'_j \land & \forall e \in \xi.E. w \xrightarrow{po} e \xrightarrow{po} t'_j \Rightarrow (\text{loc}(e) \neq x \lor e \notin W)) \\
\land (w \notin \xi.E) \Rightarrow (\forall e \in \xi.E. (e \xrightarrow{po} t'_j \Rightarrow (\text{loc}(e) \neq x \lor e \notin W)) & \land (w, \xi, rs_{x}) \in G.rt)
\end{cases}$$
We are now in a position to demonstrate the soundness of our implementation. Given an RA-consistent execution graph $G$ of the implementation, we construct an SI execution graph $G'$ as follows and demonstrate that si-consistent($G'$) holds.

- $G'.E = \bigcup_{\xi \in TX} \theta'_\xi.E$, with the $\text{tx(.)}$ function defined as:
  \[
  \text{tx}(a) \triangleq \xi \quad \text{where} \quad a \in \theta'_\xi.
  \]
- $G'.po = G.po|_{G'.E}$
- $G'.rf = \bigcup_{\xi \in TX} \text{RF}_\xi$
- $G'.mo = G.mo|_{G'.E}$
- $G'.lo = \emptyset$

Observe that the events of each $\theta'_\xi$ trace coincides with those of the equivalence class $T_\xi$ of $G'$. That is, $\theta'_\xi.E = T_\xi$.

**Lemma 3.** Given an RA-consistent execution graph $G$ of the implementation and its corresponding SI execution graph $G'$ constructed as above, for all $a, b, \xi_a, \xi_b, x$:

\[
\begin{align*}
\xi_a \neq \xi_b \land a \in \xi_a.E & \land b \in \xi_b.E \land \text{loc}(a) = \text{loc}(b) = x \implies \\
((a, b) \in G'.rf & \implies \xi_a.wu_x \xrightarrow{G.bb} \xi_b.rl_x) \quad (14) \\
\land ((a, b) \in G'.mo & \implies \xi_a.wu_x \xrightarrow{G.bb} \xi_b.rl_x) \quad (15) \\
\land ((a, b) \in G'.mo; rf & \implies \xi_a.wu_x \xrightarrow{G.bb} \xi_b.rl_x) \quad (16) \\
\land ((a, b) \in G'.rb & \implies ((x \in WS_{\xi_a} \land \xi_a.wu_x \xrightarrow{G.bb} \xi_b.rl_x) \lor (x \notin WS_{\xi_a} \land \xi_a.ru_x \xrightarrow{G.bb} \xi_b.pl_x) )) \quad (17)
\end{align*}
\]

**Proof.** Pick an arbitrary RA-consistent execution graph $G$ of the implementation and its corresponding SI execution graph $G'$ constructed as above. Pick an arbitrary $a, b, \xi_a, \xi_b, x$ such that $\xi_a \neq \xi_b, a \in \xi_a.E, a \in \xi_a.E$, and $\text{loc}(a) = \text{loc}(b) = x$.

**RTS. (14)**
Assume $(a, b) \in G'.rf$. From the definition of $G'.rf$ we then know $(a, \xi_a.rs_x) \in G.rf$. On the other hand, from Lemma 2 we know that either i) $x \in WS_{\xi_a}$ and $\xi_b.wu_x \xrightarrow{G.bb} \xi_a.rl_x$; or ii) $x \notin WS_{\xi_a}$ and $\xi_b.ru_x \xrightarrow{G.bb} \xi_a.pl_x$; or iii) $\xi_a.wu_x \xrightarrow{G.bb} \xi_b.rl_x$. In case (i) we then have $a \xrightarrow{G.bb} \xi_a.rs_x \xrightarrow{G.go} \xi_b.wu_x \xrightarrow{G.bb} \xi_a.rl_x \xrightarrow{G.go} a$. That is, we have $a \xrightarrow{G.bb} a$, contradicting the assumption that $G$ is RA-consistent. Similarly in case (ii) we have $a \xrightarrow{G.bb} \xi_a.rs_x \xrightarrow{G.go} \xi_b.ru_x \xrightarrow{G.bb} \xi_a.pl_x \xrightarrow{G.go} a$. That is, we have $a \xrightarrow{G.bb} a$, contradicting the assumption that $G$ is RA-consistent. In case (iii) the desired result holds trivially.

**RTS. (15)**
Assume $(a, b) \in G'.mo$. From the definition of $G'.mo$ we then know $(a, b) \in G.mo$.
On the other hand, from Lemma 2 we know that either i) \( \xi_b.wu_x G_{bb}^b \xi_a.rl_x \); or ii) \( \xi_a.wu_x G_{bb}^b \xi_b.rl_x \). In case (i) we then have \( a \mapsto b \mapsto _{G_{po}^a} b \mapsto _{G_{bb}^b} \xi_b.wu_x \mapsto _{G_{po}^a} \xi_a.rl_x \mapsto _{G_{po}^a} a \). That is, we have a \( a \mapsto \), contradicting the assumption that \( G \) is RA-consistent. In case (ii) the desired result holds trivially.

**RTS. (16)**

Assume \( (a, b) \in G'.mo; rf \). We then know there exists \( w \) such that \( (a, w) \in G'.mo \), \( (w, b) \in G'.rf \). From the definition of \( G'.mo \) we then know \( (a, w) \in G.mo \). There are now three cases to consider: 1) \( w \in \xi_a \); or 2) \( w \in \xi_b \); or 3) \( w \in \xi_c \wedge \xi_c \neq \xi_b \). In case (1) the desired result follows from part 14. In case (2) since \( (a, w) \in G'.mo \) the desired result follows from part 15.

In case (3) from the proof of part 15 we have \( \xi_a.wu_x G_{bb}^b \xi_c.rl_x \). Moreover, from the shape of \( G \) traces we have \( \xi_c.rl_x G_{bb}^b \xi_c.wu_x \). On the other hand, from the proof of part 14 we have \( \xi_a.wu_x G_{bb}^b \xi_b.rl_x \). We thus have \( \xi_a.wu_x G_{bb}^b \xi_c.rl_x G_{bb}^b \xi_c.wu_x G_{bb}^b \xi_b.rl_x \). As \( G.po \subseteq G hb \) and \( G hb \) is transitively closed, we have \( \xi_a.wu_x G_{bb}^b \xi_b.rl_x \), as required.

**RTS. (17)**

Assume \( (a, b) \in G'.rb \). From the definition of \( G'.rb \) we then know \( (\xi_a.rs_x, b), (\xi_a.vs_x, b) \in G.rb \). On the other hand, from Lemma 2 we know that either i) \( \xi_b.wu_x G_{bb}^b \xi_a.rl_x \); or ii) \( x \notin WS_{\xi_a} \) and \( \xi_a.rs_x G_{bb}^b \xi_a.pl_x \); or iii) \( x \in WS_{\xi_a} \) and \( \xi_a.wu_x G_{bb}^b \xi_b.rl_x \). In case (i) we then have \( b \mapsto _{G_{po}^b} b \mapsto _{G_{bb}^b} \xi_b.wu_x \mapsto _{G_{bb}^b} \xi_a.rl_x \mapsto _{G_{po}^b} b \). That is, we have \( b \mapsto \), contradicting the assumption that \( G \) is RA-consistent. In cases (ii-iii) the desired result holds trivially.

**Lemma 4.** For all RA-consistent execution graphs \( G \) of the implementation and their counterpart SI execution graphs \( G' \) constructed as above,

1. \((G'.po_T \subseteq G.po) \wedge (G'.po_T; G'.rb_T \subseteq G.hb)\)
2. \((G'.mo_T \subseteq G.hb) \wedge (G'.mor_T; G'.rb_T \subseteq G.hb)\)
3. \((G'.rf_T \subseteq G.hb) \wedge (G'.rf_T; G'.rb_T \subseteq G.hb)\)

**Proof.** Pick an arbitrary RA-consistent execution graph \( G \) of the implementation and its counterpart SI execution graph \( G' \) constructed as above.

**RTS. (Part 1)**

The proof of the first conjunct is immediate from the definitions of \( G'.po \) and \( G.po \). For the second conjunct, pick arbitrary \((a, b) \in G'.(po_T; rb_T)\). We then know there exist \( c \) such that \((a, c) \in G'.po_T \) and \((c, b) \in G'.rb_T \). Since \((a, c) \in G'.po_T \), from the definition of \( G'.po \) we also have \((a, c) \in G.po \) and thus \((a, c) \in G.hb \). Moreover, from the definition of \( G'.rb_T \) we know there exist \( \xi_1, \xi_2, r, w \) such that \( \xi_1 \neq \xi_2 \), \( r, w \in \theta_{\xi_2} \), \( b, w \in RS_{\xi_2} \) and \((r, w) \in G'.rb \). Let \( \text{loc}(r) = \text{loc}(w) = x \). We then know that \( x \in RS_{\xi_1} \cup WS_{\xi_1} \), \( x \in WS_{\xi_2} \), and that there exists \( w_x \) such that
That is, as \( G \)(i), from the proof of the first conjunct recall that we have \( (w_x, \xi_1, r, s_x) \in G^{.rf} \) and \( (w_x, w) \in G^{.mo} \). From the construction of \( G^{.mo} \) we then have \( (w_x, w) \in G^{.mo} \). From the construction of \( G^{.rf} \) there are now two cases to consider: 1) \( (w_x, \xi_1, r, s_x) \in G^{.rf} \); or 2) \( x \in WS_{\xi_1}, w_x \in \xi_1 \) and \( (w_x, r) \in G^{.po} \).

In case (1) we then have \( (\xi_1, r, s_x, w) \in G^{.rb} \). Moreover, from Lemma 3 we have that either: i) \( x \notin WS_{\xi_1} \land \xi_1, r, s_x \rightarrow G^{.hb} \xi_2, pl_x \); or ii) \( x \in WS_{\xi_1} \land \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_x \).

In case (1.i), since we have \( (a, c) \in G^{.po} \), from the construction of \( G^{.po} \) we have \( (a, \xi_1, ru_x) \in G^{.po} \). We thus have \( \xi_1, ru_x \rightarrow G^{.hb} \xi_2, pl_x \rightarrow G^{.po} b \). That is, as \( G^{.po} \subseteq G^{.hb} \) and \( G \) is transitively closed, we have \( (a, b) \in G^{.hb} \), as required.

Similarly, in case (1.ii), since we have \( (a, c) \in G^{.po} \), from the construction of \( G^{.po} \) we have \( (a, \xi_1, ru_x) \in G^{.po} \). We thus have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_x \rightarrow G^{.po} b \).

That is, as \( G^{.po} \subseteq G^{.hb} \) and \( G \) is transitively closed, we have \( (a, b) \in G^{.hb} \), as required.

In case (2), from Lemma 3 we have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_x \). Moreover, we have \( (c, \xi_1, w_{ux}) \in G^{.po} \) and \( (\xi_2, rl_x, b) \in G^{.po} \). We thus have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_x \rightarrow G^{.po} b \). That is, as \( G^{.po} \subseteq G^{.hb} \) and \( G \) is transitively closed, we have \( (a, b) \in G^{.hb} \), as required.

RTS. (Part 2)

For the first conjunct, pick an arbitrary \( (a, b) \in G^{.mo} \); we are then required to show that \( (a, b) \in G^{.hb} \).

From the definition of \( G^{.mo} \) and the construction of \( G \) we know there exist \( \xi_1, \xi_2, d, e \) such that \( \xi_1 \neq \xi_2 \), \( (d, e) \in G^{.mo} \), \( a, d \in T_{\xi_1} \) and \( b, e \in T_{\xi_2} \). Let \( \text{loc}(d) = \text{loc}(e) = y \). We then know \( \xi_1, w_{ux} \rightarrow G^{.po} \xi_1, rl_y \rightarrow G^{.po} d \rightarrow G^{.po} \xi_1, w_{uy} \rightarrow G^{.po} \xi_2, rl_x \rightarrow G^{.po} b \) and \( \xi_2, rl_y \rightarrow G^{.po} e \rightarrow G^{.po} \xi_2, w_{uy} \).

From Lemma 3 we then know \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_y \). As such, we have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_y \rightarrow b \). As \( G^{.po} \subseteq G^{.hb} \) and \( G^{.hb} \) is transitively closed, we have \( \xi_1, w_{ux} \rightarrow G^{.hb} b \), as required.

For the second conjunct, pick an arbitrary \( c \) such that \( (b, c) \in G^{.rb} \). We are then required to show that \( (a, c) \in G^{.hb} \). From the definition of \( G^{.rb} \) we then know there exist \( \xi_3, r, w \) such that \( \xi_3 \neq \xi_2 \), \( r \in T_{\xi_2} \), \( c, w \in T_{\xi_3} \) and \( (r, w) \in G^{.rb} \). Let \( \text{loc}(r) = \text{loc}(w) = x \). We then know that \( x \in RS_{\xi_2} \cup WS_{\xi_3}, x \in WS_{\xi_3} \), and that there exists \( w_x \) such that \( (w_x, r) \in G^{.rf} \) and \( (w_x, w) \in G^{.mo} \). From the construction of \( G^{.mo} \) we then have \( (w_x, w) \in G^{.mo} \). From the construction of \( G^{.rf} \) there are now two cases to consider: 1) \( (w_x, \xi_2, r, s_x) \in G^{.rf} \); or 2) \( x \in WS_{\xi_2}, w_x \in \xi_2 \) and \( (w_x, r) \in G^{.po} \).

In case (1) we then have \( (\xi_2, r, s_x, w) \in G^{.rb} \). Moreover, from Lemma 3 we have either i) \( x \notin WS_{\xi_2} \land \xi_2, ru_x \rightarrow G^{.hb} \xi_3, pl_x \); or ii) \( x \in WS_{\xi_2} \land \xi_2, ru_x \rightarrow G^{.hb} \xi_3, rl_x \). In case (1.i), from the proof of the first conjunct recall that we have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_y \). Also, from the shape of \( G \) traces we know that \( \xi_2, rl_y \rightarrow G^{.po} \xi_2, ru_x \).

As such, we have \( \xi_1, w_{ux} \rightarrow G^{.hb} \xi_2, rl_y \rightarrow G^{.po} \xi_2, ru_x \rightarrow G^{.hb} \xi_3, pl_x \rightarrow G^{.hb} c \). That is, as \( G^{.po} \subseteq G^{.hb} \) and \( G \) is transitively closed, we have \( (a, c) \in G^{.hb} \), as required.
Similarly, in case (1.ii), from the proof of the first conjunct recall that we have \( G^{\po} \xrightarrow{\xi_1.\wu_y} G^{\bb} \xrightarrow{\xi_2.\rl_y}. \) Also, from the shape of \( G \) traces we know that \( \xi_2.\rl_y \xrightarrow{G^{\po}} \xi_2.\wu_x \). As such, we have \( a \xrightarrow{G^{\po}} \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \xrightarrow{G^{\po}} \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \xrightarrow{G^{\po}} c. \) That is, as \( G^{\po} \subseteq G^{\hb} \) and \( G \) is transitively closed, we have \((a,c) \in G^{\hb}\), as required.

In case (2) from Lemma 3 we have \( \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \). Recall that with the first conjunct we demonstrated that \((a,b) \in G^{\hb}\). Moreover, we have \( (b, \xi_2.\wu_x) \in G^{\po} \) and \(( \xi_3.\rl_x, c) \in G^{\po}\). We thus have, \( a \xrightarrow{G^{\bb}} b \xrightarrow{G^{\po}} \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \xrightarrow{G^{\po}} c. \) That is, as \( G^{\po} \subseteq G^{\hb} \) and \( G \) is transitively closed, we have \((a,b) \in G^{\hb}\), as required.

RTS. (Part 3)
For the first conjunct, pick an arbitrary \((a,b) \in G^{\prime}.rF_T\); we are then required to show that \((a,b) \in G^{\hb}\).

From the definition of \( G^{\prime}.rF_T \) and the construction of \( G^{\prime} \) we know there exist \( \xi_1, \xi_2, w_y, r_y \) such that \( \xi_1 \neq \xi_2 \), \( (w_y, r_y) \in G^{\prime}.rF \), \( a, w_y \in T_1 \) and \( b, r_y \in T_2 \). Let \( 1\text{oc}(w) = 1\text{oc}(r) = y \). We then know \( \xi_1.\rl_y \xrightarrow{G^{\po}} \xi_1.\wu_y \xrightarrow{G^{\bb}} w_y \xrightarrow{G^{\po}} \xi_1.\wu_y \) and \( a \xrightarrow{G^{\po}} \xi_1.\wu_y \).

Let \( w_y = \Psi(y, v) \) and \( r_y = R(y, v) \). From the construction of \( G^{\prime} \) we know that \((w_y, \xi_2.\rs_y) \in G^{\prime}.rF \). On the other hand, from Lemma 3 we have \( \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \). We then have \( a \xrightarrow{G^{\bb}} \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \xrightarrow{G^{\po}} b. \) That is, as \( G^{\po} \subseteq G^{\hb} \), we have \( a \xrightarrow{G^{\bb}} b \), as required.

For the second conjunct, pick an arbitrary \( c \) such that \((b,c) \in G^{\prime}.rB_T \). We are then required to show that \((a,c) \in G^{\hb}\). From the definition of \( G^{\prime}.rB_T \) we then know there exist \( \xi_3, r, w \) such that \( \xi_3 \neq \xi_2, r \in T_{r_3}, c, w \in T_{r_3}, \) and \( (r, w) \in G^{\prime}.rB \). Let \( 1\text{oc}(r) = 1\text{oc}(w) = x. \) We then know that \( x \in R^\prime \xi_2 \cup S^\prime \xi_3^2 \), \( x \in S^\prime \xi_3 \), and that there exists \( w_x \) such that \( (w_x, r) \in G^{\prime}.rF \) and \( (w_x, w) \in G^{\prime}.mo \). From the construction of \( G^{\prime}.mo \) we then have \((w_x, w) \in G^{\prime}.mo \). From the proof of the first conjunct recall that we have \( a \xrightarrow{G^{\po}} \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \).

From the construction of \( G^{\prime}.rF \) there are now two cases to consider: 1) \((w_x, \xi_2.\rs_x) \in G^{\prime}.rF \); or 2) \( x \in WS^\prime \xi_3 \), \( w_x \in T_3 \) and \((w_x, r) \in G^{\prime}.po \). In case (1) we then have \((\xi_2.\rs_x, w) \in G^{\prime}.rB \). Moreover, from Lemma 3 we have either: i) \( \xi_2.\rs_x \xrightarrow{G^{\bb}} \xi_3.\pl_x \) or ii) \( \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \).

In case (1.i), from the shape of \( G \) traces we know that \( \xi_2.\rl_x \xrightarrow{G^{\po}} \xi_2.\wu_x \). As such, we have \( a \xrightarrow{G^{\po}} \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \xrightarrow{G^{\po}} \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \xrightarrow{G^{\po}} c. \) That is, as \( G^{\po} \subseteq G^{\hb} \) and \( G \) is transitively closed, we have \((a,c) \in G^{\hb}\), as required.

Similarly, in case (1.ii), from the shape of \( G \) traces we know that \( \xi_2.\rl_y \xrightarrow{G^{\po}} \xi_2.\wu_x \). As such, we have \( a \xrightarrow{G^{\po}} \xi_1.\wu_y \xrightarrow{G^{\bb}} \xi_2.\rl_y \xrightarrow{G^{\po}} \xi_2.\wu_x \xrightarrow{G^{\bb}} \xi_3.\rl_x \xrightarrow{G^{\po}} c. \) That is, as \( G^{\po} \subseteq G^{\hb} \) and \( G \) is transitively closed, we have \((a,c) \in G^{\hb}\), as required.
In case (2), from Lemma 3 we have $\xi_2.wu_x G^G hb \rightarrow \xi_3.rl_x$. Recall that with the first conjunct we demonstrated that $(a,b) \in G.hb$. Moreover, we have $(b, \xi_2.wu_x) \in G.po$ and $(\xi_3.rl_x, c) \in G.po$. We thus have, $a G^G hb b G^G po \xi_2.wu_x G^G hb \xi_3.rl_x G^G po c$. That is, as $G.po \subseteq G.hb$ and $G$ is transitively closed, we have $(a,b) \in G.hb$, as required.

**Theorem 7 (Soundness).** For all execution graphs $G$ of the implementation and their counterpart SI execution graphs $G'$ constructed as above,

$$\text{RA-consistent}(G) \Rightarrow \text{si-consistent}(G')$$

**Proof.** Pick an arbitrary execution graph $G$ of the implementation such that $\text{RA-consistent}(G)$ holds, and its associated SI execution graph $G'$ constructed as described above.

**RTS. acyclic** $(G', (po_\mathcal{T} \cup rf_\mathcal{T} \cup mo_\mathcal{T}); rb_\mathcal{T}^-))$

We proceed by contradiction. Let us assume

$$\text{RA-consistent}(G) \land \neg \text{acyclic}(G', ((po_\mathcal{T} \cup rf_\mathcal{T} \cup mo_\mathcal{T}); rb_\mathcal{T}^-)) \quad (18)$$

From the definition of $\text{RA-consistent}(\cdot)$ we then know that there exists a such that $(a,a) \in ((G'.po_\mathcal{T} \cup G'.rf_\mathcal{T} \cup G'.mo_\mathcal{T}); G'.rb_\mathcal{T}^-)^T$. Consequently, from Lemma 4 we have $(a,a) \in G.hb^+$. That is, since $G.hb$ is transitively closed, we have $(a,a) \in G.hb$, contradicting our assumption that $G$ is RA-consistent.

**RTS. rf_\mathcal{T} \cup mo_\mathcal{T} \cup rb_\mathcal{T} \subseteq po_\mathcal{T}$$

Follows immediately from the construction of $G'$.

### C.2 Implementation Completeness

In order to establish the completeness of our implementation, it suffices to show that given an SI-consistent execution graph $G' = (E, po, rf, mo, lo)$, we can construct a corresponding RA-consistent execution graph $G$ of the implementation. Before proceeding with the construction of a corresponding implementation graph, we describe several auxiliary definitions.

Given an abstract transaction class $T_\xi \in G'.T/st$, we write $\mathcal{W}_\xi$ for the set of locations written to by $T_\xi$: $\mathcal{W}_\xi = \bigcup_{e \in T_\xi} \mathcal{W}(e)$. Similarly, we write $\mathcal{R}_\xi$ for the set of locations read from by $T_\xi$, prior to being written by $T_\xi$. For each location $x$ read from by $T_\xi$, we additionally record the first read event in $T_\xi$ that retrieved the value of $x$. That is,

$$\mathcal{R}_T \xi \triangleq \{ (x, r) \mid r \in T_\xi \cap R_x \land \exists e \in T_\xi \land E_x \land e \rightarrow_p r \}$$

Note that the execution trace for each transaction $T_\xi \in G'.T/st$ is of the form $\theta_\xi = t'_1 \circ \ldots \circ t'_k$, for some $k$, where each $t'_i$ is a read or write event. As such, we have $G'.E = G'.T = \bigcup_{T_\xi \in G'.T/st} T_\xi = \theta_\xi$.E.
Let \( RS_{\xi} \cup WS_{\xi} = \{ x_1 \cdots x_n \} \). We then construct the implementation trace \( \theta_{\xi} \) as:

\[
\theta_{\xi} = Rs \xrightarrow{po|imm} RU_{\xi} \xrightarrow{po|imm} PL_{\xi} \xrightarrow{po|imm} T_{\xi} \xrightarrow{po|imm} U_{\xi}
\]

where

\( - \quad Rs = rl_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} rl_{x_n} \xrightarrow{po|imm} S_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} S_{x_n} \), where the identifiers of all constituent events of \( Rs \) are picked fresh, and

\[
rl_{x_j} = RL(x_j)
\]

\[
S_{x_j} = \begin{cases} 
rs_{x_j} \xrightarrow{po|imm} ws_{x_j} & \text{if } \exists r. (x_j, r) \in RS_{\xi} \land val(r) = v_j \\
\emptyset & \text{otherwise}
\end{cases}
\]

with \( rs_{x_j} \triangleq R(x_j, v_j) \) and \( ws_{x_j} \triangleq W(s[x_j], v_j) \).

\( - \quad RU_{\xi} = ru_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} ru_{x_n} \), where the identifiers of all constituent events of \( RU_{\xi} \) are picked fresh, and for all \( j \in \{1 \cdots n\} \):

\[
ru_{x_j} = \begin{cases} 
RU(x_j) & \text{if } x_j \notin WS_{\xi} \\
\emptyset & \text{otherwise}
\end{cases}
\]

\( - \quad PL_{\xi} = pl_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} pl_{x_n} \), where the identifiers of all constituent events of \( PL_{\xi} \) are picked fresh, and for all \( j \in \{1 \cdots n\} \):

\[
pl_{x_j} = \begin{cases} 
PL(x_j) & \text{if } x_j \in WS_{\xi} \\
\emptyset & \text{otherwise}
\end{cases}
\]

\( - \quad T_{\xi} = t_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t_k \), where for all \( m \in \{1 \cdots k\} \):

\[
t_m = \begin{cases} 
R(s[x_n], v_n) & \text{if } t'_m = R(x_n, v_n) \\
W(x_n, v_n) \xrightarrow{po|imm} W(s[x_n], v_n) & \text{if } t'_m = W(x_n, v_n)
\end{cases}
\]

such that in the first case the identifier of \( t_m \) is that of \( t'_m \); and in the second case the identifier of the first event in \( t_m \) is that of \( t'_m \) and the identifier of the second event is picked fresh.

\( - \quad U_{\xi} = wu_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} wu_{x_n} \), where the identifiers of all constituent events of \( U_{\xi} \) are picked fresh, and

\[
wu_{x_j} = \begin{cases} 
WU(x_j) & \text{if } x_j \in WS_{\xi} \\
\emptyset & \text{otherwise}
\end{cases}
\]

We use the \( \xi \) prefix to project the various events of the implementation trace \( \theta_{\xi} \) (e.g. \( \xi.rl_{x_j} \)).
Given the transaction classes \( T/st \) of \( G' \), we construct a strict total order \( TCO : T/st \times T/st \) as an extension of \( G'.(rb_T^2; (po_T \cup mo_T \cup rf_T))^+ \). That is, construct \( TCO \) as a total order such that:

\[
\forall e, e', (e,e') \in G'.(rb_T^2; (po_T \cup mo_T \cup rf_T))^+ \Rightarrow ([e]_{st} , [e']_{st} ) \in TCO
\]

Recall that since \( G' \) is SI-consistent, we know \( \text{acyclic}((po_T \cup mo_T \cup rf_T); rb_T^2) \) holds, i.e. \( \text{acyclic}(rb_T^2; (po_T \cup mo_T \cup rf_T)) \). As such, it is always possible to extend \( (rb_T^2; (po_T \cup mo_T \cup rf_T))^+ \) to a total order as described above.

For each location \( x \), let \( WT_x \triangleq \{ T \in T/st | W_x \cap T \neq \emptyset \} \) denote those transactions that write to \( x \). We then define \( \text{TMO}_x \triangleq TCO|_{WT_x} \), a strict total order on \( WT_x \). Given a strict total order \( r \), we write \( r_i \) for the \( i \)th item of \( r \), indexed from 0 (e.g. \( \text{TMO}_x[i] \)). For \( i \in \mathbb{N} \), we then define:

\[
\begin{align*}
S^0_x \triangleq & \{ T_{\ell} \mid \text{TMO}_x[0] = T_{\ell} \lor \exists r \in T_{\ell}, (\text{init}_x, r) \in G'.rf \} \\
S^i_{x+1} \triangleq & \{ T_{\ell} \mid \text{TMO}_x[i+1] = T_{\ell} \lor \exists r \in T_{\ell}, \exists w \in \text{TMO}_x[i], (w, r) \in G'.rf \setminus rf_1 \}
\end{align*}
\]

where \( \text{init}_x \) denotes the write event initialising the value of \( x \) (with zero).

For all locations \( x \) and \( i \in \mathbb{N} \), let \( P^*_x \triangleq TCO|_{S^i_x} \). Note that for each \( i \in \mathbb{N} \), the \( S^i_x \) contains at most one transaction that writes to \( x \). We denote this transaction by \( P^*_x.writer \). For each lock \( x \) we then define:

\[
\begin{align*}
\text{LO}_x & \triangleq \left\{ (P^*_x.writer.pl_x, P^*_x.writer.wu_x), (P^*_x.writer.pl_x, P^*_x.kj.pl_x), (P^*_x.writer.pl_x, P^*_x.kj.wu_x), (P^*_x.writer.pl_x, P^*_x.writer.pl_x), (P^*_x.writer.pl_x, P^*_x.writer.wu_x), (P^*_x.writer.wu_x, P^*_x.kj.pl_x), (P^*_x.writer.wu_x, P^*_x.kj.wu_x), (P^*_x.writer.wu_x, P^*_x.writer.pl_x), (P^*_x.writer.wu_x, P^*_x.writer.wu_x) \right\} \\
\end{align*}
\]

\[
\begin{align*}
\cup \left\{ (P^*_x.kj.pl_x, r_l_x, P^*_x.writer.pl_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.writer.pl_x), (P^*_x.kj.pl_x, r_l_x, P^*_x.writer.wu_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.writer.wu_x) \right\} \\
\end{align*}
\]

\[
\begin{align*}
\text{RLO}_x & \triangleq \text{LO}_x \cup \left\{ (P^*_x.kj.pl_x, r_l_x, P^*_x.kj.pl_x), (P^*_x.kj.pl_x, r_l_x, P^*_x.kj.wu_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.kj.pl_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.kj.wu_x) \right\} \\
\end{align*}
\]

\[
\begin{align*}
\cup \left\{ (P^*_x.kj.pl_x, r_l_x, P^*_x.kj.pl_x), (P^*_x.kj.pl_x, r_l_x, P^*_x.kj.wu_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.kj.pl_x), (P^*_x.kj.pl_x, r_u_x, P^*_x.kj.wu_x) \right\} \\
\end{align*}
\]

Remark 4. Let \( \text{lo}_1 \triangleq \bigcup_{x \in \text{LOC}} \text{LO}_x \) and \( \text{lo}_2 \triangleq \bigcup_{x \in \text{LOC}} \text{RLO}_x \). Note that both \( \text{lo}_1 \) and \( \text{lo}_2 \) satisfy the conditions stated in Def. 4. The \( \text{lo}_2 \) additionally satisfies the
'read-read-synchronisation' property in (RSYNC). In what follows we demonstrate that given an SI-consistent execution graph \( G' \), it is always possible to construct an RA-consistent execution graph \( G \) of the implementation with its lock order defined as \( \text{lo}_2 \). Note that as \( \text{lo}_1 \subseteq \text{lo}_2 \), it is straightforward to show that replacing \( \text{lo}_2 \) in such a \( G \) with \( \text{lo}_1 \), preserves the RA-consistency of \( G \), as defined in Def. 4.

In other words, as \( \text{lo}_1 \subseteq \text{lo}_2 \), we have:

\[
\text{acyclic}(\text{hb}_{\text{loc}} \cup \text{mo} \cup \text{rb}) \text{ with } \text{hb} \equiv (\text{po} \cup \text{rf} \cup \text{lo}_2)^+ \Rightarrow \\
\text{acyclic}(\text{hb}_{\text{loc}} \cup \text{mo} \cup \text{rb}) \text{ with } \text{hb} \equiv (\text{po} \cup \text{rf} \cup \text{lo}_1)^+
\]

As such, by establishing the completeness of our implementation with respect to \( \text{lo}_2 \), we also establish its completeness with respect to \( \text{lo}_1 \). In other words, we demonstrate the completeness of our implementation with respect to both lock implementations presented earlier in Appendix A.

We now demonstrate the completeness of our implementation. Given an SI-consistent graph, we construct an implementation graph \( G \) as follows and demonstrate that it is RA-consistent.

- \( G.E = \bigcup_{T \in T_G. T} \theta_T.E \), with the \( \text{tx}(e) = 0 \), for all \( e \in G.E \).

Observe that \( G'.E \subseteq G.E \).
- \( G.E = \bigcup_{T \in T_G. E} \theta_T.E \), with the \( \text{tx}(e) = 0 \), for all \( e \in G.E \).

Observe that \( G'.E \subseteq G.E \).
- \( G.E = \bigcup_{T \in T_G. E} \theta_T.E \), with the \( \text{tx}(e) = 0 \), for all \( e \in G.E \).

Observe that \( G'.E \subseteq G.E \).
- \( G.E = \bigcup_{T \in T_G. E} \theta_T.E \), with the \( \text{tx}(e) = 0 \), for all \( e \in G.E \).

Observe that \( G'.E \subseteq G.E \).

**Notation**

Given an implementation graph \( G \) as constructed above (with \( G.T = \emptyset \)), and a relation \( r \subseteq G.E \times G.E \), we override the \( r_E \) notation and write \( r_E \) for:

\[
\{(a, b) \in r | \exists \xi_a, \xi_b. a \in \xi_a.E \land b \in \xi_b.E \land \xi_a \neq \xi_b\}
\]

Analogously, we write \( r_I \) for \( \{(a, b) \in r \mid \exists \xi. a, b \in \xi.E\} \).

**Lemma 5.** For all SI-consistent execution graphs \( G' = (E, \text{po}, \text{rf}, \text{mo}, \text{lo}) \), and for all \( x \in \text{Loc}, \xi_a, \xi_b \in \text{TXIn}, \) and \( i \in \mathbb{N} \):

\[
\xi_a \in S_i^x \land \xi_b \in S_{i+1}^x \Rightarrow (\xi_a, \xi_b) \in \text{TCO}
\]

**Proof.** Pick arbitrary SI-consistent execution graph \( G' = (E, \text{po}, \text{rf}, \text{mo}, T) \), \( x \in \text{Loc}, \xi_a, \xi_b \in \text{TXIn}, \) and \( i \in \mathbb{N} \). There are now four cases to consider: 1) \( \xi_a = P_i^x.\text{writer} \) and \( \xi_b = P_{i+1}^x.\text{writer} \); 2) \( \xi_a = P_i^x.\text{writer} \) and \( \xi_b \neq P_{i+1}^x.\text{writer} \); 3) \( \xi_a \neq P_i^x.\text{writer} \) and \( \xi_b = P_{i+1}^x.\text{writer} \); and 4) \( \xi_a \neq P_i^x.\text{writer} \) and \( \xi_b \neq P_{i+1}^x.\text{writer} \).

In case (1) from the definitions of \( S_i^x \) and \( P_i^x.\text{writer} \) we have \( T_{\xi_a} \times T_{\xi_b} \subseteq \text{mo}\), and thus from the definition of TCO we have \( (\xi_a, \xi_b) \in \text{TCO} \), as required. Similarly,
in case (2) from the definitions of $S^x_i$ and $P^x_i$ we have $T_{E_x} \times T_{E_b} \subseteq \subseteq r f_T$
and thus from the definition of TCO we have $(\xi_a, \xi_b) \in \text{TCO}$, as required.

In case (3), from the definition of $S^x_i$ we know there exist $r \in T_{E_x} \cap R_x$ and $w \in P^x_i$ such that $(r, w) \in r f_b$ and thus $(a, w) \in r f_T$. On the other hand from the definitions of $S^x_i$ and $S^x_{i+1}$, we know there exists $w' \in T_{E_b} \cap W_x$ such that $(w, w') \in m o$ and thus $(a, w') \in r f_T; m o_T$. As such we have $(a, w') \in r f_T; m o_T$
and thus from the definition of TCO we have $(\xi_a, \xi_b) \in \text{TCO}$, as required.

In case (4), from the definition of $S^x_i$ we know there exist $r \in T_{E_x} \cap R_x$ and $w \in P^x_i$ such that $(r, w) \in r f_b$ and thus $(a, w) \in r f_T$. On the other hand from the definitions of $S^x_i$ and $S^x_{i+1}$, we know there exists $r' \in T_{E_b} \cap R_x$ such that $(w, r') \in r f_b$ and thus $(w, r') \in r f_T$. As such we have $(a, r') \in r f_T; r f_T$ and thus from the definition of TCO we have $(\xi_a, \xi_b) \in \text{TCO}$, as required.

**Corollary 1.** For all SL-consistent execution graphs $G' = (E, p o, r f, m o, l o)$, and for all $x \in L o c, \xi_a, \xi_b \in \text{TXIn}$, and $i, j \in N$:

$$
\xi_a \in S^x_i \wedge \xi_b \in S^x_j \land i < j \implies (\xi_a, \xi_b) \in \text{TCO}
$$

**Proof.** Follows by induction from Lemma 5.

Given an implementation graph $G = (E, p o, r f, m o, l o)$, let

$$
b^b' = (E, p o \cup r f \cup l o) \iff \exists i, j, k. P^x_i|_j = \xi_1 \land P^x_j|_k = \xi_2 \land j > k
$$

**Lemma 6.** For all implementation graphs $G = (E, p o, r f, m o, l o)$ constructed as above,

$$
\forall a, b, \xi_a, \xi_b. a \in \theta_{E_x}. E \wedge b \in \theta_{E_b}. E \wedge a \overset{b^b'}{\rightarrow} b
\implies (a, b) \in G.p o \lor (\xi_a, \xi_b) \in \text{TCO}
$$

**Proof.** Pick an arbitrary implementation graph $G = (E, p o, r f, m o, l o)$ constructed as above. Since $b^b'$ is a transitive implementation graph, it is straightforward to demonstrate that $b^b' = \bigcup_{i \in N} b^b'_i$, where:

$$
b^b'_0 = \{ (\xi_1, r l_x, \xi_2, p l_x) \in \text{lo}_x, (\xi_1, r u_x, \xi_2, p u_x) \in \text{lo}_x, \}
$$

and $b^b'_{i+1} = b^b'_i; b^b'_{i+1}$. It thus suffices to show:

$$
\forall i \in N, \forall a, b, \xi_a, \xi_b. a \in \theta_{E_x}. E \wedge b \in \theta_{E_b}. E \wedge a \overset{b^b'_i}{\rightarrow} b
\implies (a, b) \in G.p o \lor (\xi_a, \xi_b) \in \text{TCO}
$$

We proceed by induction over $i$. 
Base case $i = 0$

Pick arbitrary $a, b, \xi_a, \xi_b$ such that $a \in \theta_{\xi_a} E$ and $b \in \theta_{\xi_b} E$, $a \xrightarrow{\text{hb}'} b$. There are then four cases to consider: 1) $a \xrightarrow{\text{g.po}_1 (\text{rf})} b$; or 2) $a \xrightarrow{\text{g.po}_2} b$; 3) $a \xrightarrow{\text{g.f}_E} b$; or 4) $a \xrightarrow{\text{g.lo}_2} b$.

In case (1), from the construction of $G.\text{rf}_1$ we have $G.\text{rf}_1 \subseteq G.\text{po}_1$; moreover, from the construction of $G.\text{lo}_1$ we have $G.\text{lo}_1 \subseteq G.\text{po}_1$. Consequently we have $a \xrightarrow{G.\text{po}_1} b$, as required.

In case (2) from the construction of $G.\text{po}_2$ we then know that $T_{\xi_a} \times T_{\xi_b} \subseteq G'.\text{po}$ and thus from the definition of $\text{TCO}$ we have $\langle \xi_a, \xi_b \rangle \in \text{TCO}$, as required.

In case (3) from the construction of $G.\text{rf}$ we then know that $T_{\xi_a} \times T_{\xi_b} \subseteq G'.\text{rf}_T$ and thus from the definition of $\text{TCO}$ we have $\langle \xi_a, \xi_b \rangle \in \text{TCO}$, as required.

In case (4) from the definition of $\text{hb}'_i$ we know that there exists $i, j, i', j'$ such that $\xi_a = P'_i |_j, \xi_b = P'_i |_{j'}$ and either i) $i < i'$; or ii) $i = i' \land j < j'$.

In case (4.i) from Corollary 1 we have $\langle \xi_a, \xi_b \rangle \in \text{TCO}$, as required. In case (4.ii) from the definitions of $P'_i$ we have $\langle \xi_a, \xi_b \rangle \in \text{TCO}$, as required.

Inductive case $i = n+1$

Pick arbitrary $a, b, \xi_a, \xi_b$ such that $a \in \theta_{\xi_a} E$, $b \in \theta_{\xi_b} E$ and $a \xrightarrow{\text{hb}'} b$.

$$\forall j \leq n. \forall c, d, \xi_c, \xi_d, c \in \theta_{\xi_c} E \land d \in \theta_{\xi_d} E \land c \xrightarrow{\text{hb}'_n} d \Rightarrow (c, d) \in G.\text{po}_1 \cup \langle \xi_c, \xi_d \rangle \in \text{TCO}$$

(I.H.)

From the definition of $\text{hb}'_{n+1}$ we then know there exists $e$ such that $(a, e) \in \text{hb}'_n$ and $(e, b) \in \text{hb}'_n$. Let $e \in \xi_c E$. Consequently, from the proof of the base case we then know that $(a, e) \in G.\text{po}_1 \cup \langle \xi_a, \xi_e \rangle \in \text{TCO}$. Similarly, from (I.H.) we have $(e, b) \in G.\text{po}_1 \cup \langle \xi_c, \xi_b \rangle \in \text{TCO}$. There are now four cases to consider: 1) $(a, e) \in G.\text{po}_1$ and $(e, b) \in G.\text{po}_1$; 2) $(a, e) \in G.\text{po}_1$ and $(\xi_c, \xi_b) \in \text{TCO}$; 3) $(\xi_a, \xi_e) \in \text{TCO}$ and $(e, b) \in G.\text{po}_1$; 4) $(\xi_a, \xi_e) \in \text{TCO}$ and $(\xi_c, \xi_b) \in \text{TCO}$.

In case (1) from the definition of $G.\text{po}_1$ we have $(a, b) \in G.\text{po}_1$. In cases (2-4) from the definitions of $G.\text{po}_1$, TCO we have $\langle \xi_a, \xi_b \rangle \in \text{TCO}$, as required.

Given an implementation graph $G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo})$ constructed as above, let $\text{stg}(\_): G.E \to \{1, 2, 3\}$ denote the event stage reflecting whether it is in the snapshot phase (1), reader lock release phase (2), or the update phase (3). That is,

$$\text{stg}(e) \triangleq \begin{cases} 1 & \exists \xi, x. e \in \{\xi.\text{rl}_x, \xi.\text{rs}_x, \xi.\text{ws}_x\} \\ 2 & \exists \xi, x. e = \xi.\text{ru}_x \\ 3 & \text{otherwise} \end{cases}$$
Lemma 7. For all implementation graphs \( G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo}) \) constructed as above, and for all \( x \in \text{Loc}, \xi_\alpha, \xi_1, \xi_2 \in \text{TXId} \) and \( a \in \theta_{\xi_\alpha}E \):

\[
\begin{align*}
\left( \xi_1.rlx_{\text{lo}} & \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \right) \\
\lor (\xi_1.ru_{\text{lo}} & \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a) \\
\lor (\xi_1.rlx_{\text{lo}} & \rightarrow \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a) \\
\lor (\xi_1.ru_{\text{lo}} & \rightarrow \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a)
\end{align*}
\]  
\( \land \exists j, k, l. \ P_j^x|_k = \xi_1 \land P_j^x|_l = \xi_2 \land k > l \)

\[ \Rightarrow \text{stg}(\xi_1.ru_{\text{lo}}) < \text{stg}(a) \lor (\xi_1, \alpha) \in \text{TCO} \]

Proof. Pick an arbitrary implementation graph \( G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo}) \) constructed as above, and pick arbitrary \( x \in \text{Loc}, \alpha, \xi_1, \xi_2 \in \text{TXId} \) and \( a \in \theta_{\xi_\alpha}E \) such that 

\[(\xi_1.rlx_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a) \lor (\xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a) \lor (\xi_1.rlx_{\text{lo}} \rightarrow \\
(\xi_1.ru_{\text{lo}} \rightarrow \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a) \lor (\xi_1.ru_{\text{lo}} \rightarrow \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a)) \land \exists j, k, l. \ P_j^x|_k = \xi_1 \land P_j^x|_l = \xi_2 \land k > l.
\]

In case of the first disjunct from the construction of \( \text{lo} \) we then also have \( \xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \). Similarly, in case of the third and fourth disjuncts from the construction of \( \text{lo} \) we also have \( \xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a \).

Moreover, since we have \( \xi_2.plx_{\text{lo}} \rightarrow \xi_2.wux_{\text{x}} \land \xi_2.wux_{\text{hb}} \rightarrow a \) and \( \text{po} \subseteq \text{hb} \), we also have \( \xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \). It thus suffices to show:

\[
\xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \land \exists j, k, l. \ P_j^x|_k = \xi_1 \land P_j^x|_l = \xi_2 \land k > l
\]

\[ \Rightarrow \text{stg}(\xi_1.ru_{\text{lo}}) < \text{stg}(a) \lor (\xi_1, \alpha) \in \text{TCO} \]

Since \( G.\text{hb} \) is a transitive closure, it is straightforward to demonstrate that \( G.\text{hb} = \bigcup_{i \in \mathbb{N}} \text{hb}_i \), where \( \text{hb}_0 = G.\text{po} \cup G.\text{rf} \cup G.\text{lo} \) and \( \text{hb}_{i+1} = \text{hb}_0; \text{hb}_i \). It thus suffices to show:

\[
\forall i \in \mathbb{N}. \ \forall x \in \text{Loc}, \alpha, \xi_1, \xi_2 \in \text{TXId}, a \in \theta_{\xi_\alpha}E.
\]

\[
\xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \land \exists j, k, l. \ P_j^x|_k = \xi_1 \land P_j^x|_l = \xi_2 \land k > l
\]

\[ \Rightarrow \text{stg}(\xi_1.ru_{\text{lo}}) < \text{stg}(a) \lor (\xi_1, \alpha) \in \text{TCO} \]

We thus proceed by induction over \( i \).

Base case \( i = 0 \)

Pick arbitrary \( x \in \text{Loc}, \alpha, \xi_1, \xi_2 \in \text{TXId} \) and \( a \in \theta_{\xi_\alpha}E \) such that \( \xi_1.ru_{\text{lo}} \rightarrow \xi_2.plx_{\text{lo}} \land \xi_2.plx_{\text{hb}} \rightarrow a \) and \( \exists j, k, l. \ P_j^x|_k = \xi_1 \land P_j^x|_l = \xi_2 \land k > l \). From the construction of \( G \) we know that there exists \( r \in G'.\text{TC}_{\xi_1} \) and \( w \in G'.\text{TC}_{\xi_2} \) such that \( (r, w) \in G'.\text{rb} \).

Moreover, since \( k \neq l \) we know that \( (r, w) \in G'.\text{rb}_T \). Since we have \( \xi_2.plx_{\text{hb}} \rightarrow a \),

there are four cases to consider:

1) \( \xi_2.plx_{\text{lo}} \rightarrow G.\text{po}_{\text{lo}}T \rightarrow a \); or
2) \( \xi_2.plx_{\text{lo}} \rightarrow G.\text{po}_{\text{lo}}T \rightarrow a \); or
3) \( \xi_2.plx_{\text{lo}} \rightarrow G.\text{po}_{\text{lo}}T \rightarrow a \); or
4) \( \xi_2.plx_{\text{lo}} \rightarrow G.\text{po}_{\text{lo}}T \rightarrow a \).

In case (1), from the construction of \( G.\text{rf}_1 \) we have \( G.\text{rf}_1 \subseteq G.\text{po}_{\text{lo}} \); moreover, from the construction of \( G.\text{lo}_1 \) we have \( G.\text{lo}_1 \subseteq G.\text{po}_{\text{lo}} \). We thus have \( \xi_2.plx_{\text{lo}} \rightarrow a \).
On the other hand, from the definition of \( \text{stg}(\cdot) \) we know \( \text{stg}(a) \geq \text{stg}(\xi_2,pl_x) > \text{stg}(\xi_1,ru_x) \), and thus \( \text{stg}(a) > \text{stg}(\xi_1,ru_x) \), as required.

In case (2), since \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} a \), we have \((w,a) \in G'.p_{\xi_2}r_T\). As such, we have \((r,a) \in G'.(r_{\xi_2}r_T;p_{\xi_2}r_T)\). From the definition of TCO we thus have \((\xi_1,\xi_2) \in \text{TCO}, as required.

Case (3) cannot happen as there are no \( r_f \) edges from lock events. In case (4), from the construction of \( \text{lo} \) we know there exists \( m,n \) such that \( m > j \), and \( \xi_n = P_{m,n}^x \). As such, from Corollary 1 we have \((\xi_1,\xi_n) \in \text{TCO}, as required.

**Inductive case** \( i = n+1 \)

Pick arbitrary \( x \in \text{Loc} \) and \( \xi_n,\xi_1,\xi_2 \in \text{TXId} \) and \( a \in \theta_{\xi_n}.E \) such that \( \xi_1,ru_x \xrightarrow{lo} \xi_2,pl_x,\xi_2,pl_x \xrightarrow{hb_1} a \) and \( \exists j,k,l. \ P_j^x \equiv \xi_1 \land P_k^x \equiv \xi_2 \land k > l \).

\[ \forall j \leq n. \forall y \in \text{Loc}, \xi_{a'},\xi_3,\xi_4 \in \text{TXId}, a' \in \theta_{\xi_{a'}}.E. \]

\[ (\xi_3,ru_x \xrightarrow{lo} \xi_4,pl_x \land \xi_4,pl_x \xrightarrow{hb_2} a') \land \exists j,k,l. \ P_j^x \equiv \xi_1 \land P_k^x \equiv \xi_2 \land k > l \]

\[ \Rightarrow \text{stg}(\xi_3,ru_x) < \text{stg}(a') \lor (\xi_3,\xi_{a'}) \in \text{TCO} \]

(I.H.)

There are two cases to consider:

1) \( \xi_2,pl_x \xrightarrow{(p_{\xi_2}r_T)\cup\text{lo}_{\xi_2})} a; \) or

2) \( \xi_2,pl_x \xrightarrow{(p_{\xi_2}r_T)\cup\text{lo}_{\xi_2})}; \xi_2,pl_x \xrightarrow{hb_2} a, \) where \( m \leq n. \)

In case (1), from the construction of \( G.r_f \) we have \( G.r_f \in G.p_{\xi_1}; \) moreover, from the construction of \( G.lo \) we have \( G.lo \in G.p_{\xi_1} \). We thus have \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} a \), i.e. \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} a \). As such, from the proof of the base case we have \( \text{stg}(a) > \text{stg}(\xi_1,ru_x) \), as required.

In case (2) from the construction of \( G \) we know that there exists \( r \in \theta_{\xi_2}.E \) and \( w \in \theta_{\xi_2}.E \) such that \( (r,w) \in G'.r_f T \). Moreover, since \( k \neq l \), from the construction of \( G \) we know that \( \xi_1 \neq \xi_2 \) and thus we have \( (r,w) \in G'.r_f T \). On the other hand, we know that there exist \( b,c \) such that \( \xi_2,pl_x \xrightarrow{p_{\xi_2}r_T} b \xrightarrow{p_{\xi_2}r_T} c \xrightarrow{hb_2} a \). From the construction of \( G.r_f \) we have \( G.r_f \in G.p_{\xi_1} \); moreover, from the construction of \( G.lo \) we have \( G.lo \in G.p_{\xi_1} \). We thus have \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} b \). As such we have \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} b \xrightarrow{p_{\xi_2}r_T} c \). Let \( c \in \theta_{\xi_2} \). There are now three cases to consider: a) \( b \xrightarrow{po} c \); or b) \( b \xrightarrow{rf} c \); or c) \( b \xrightarrow{lo} c \). We first demonstrate that in all three cases we have \((\xi_1,\xi_2) \in \text{TCO} \).

In case (2.a), since \( \xi_2,pl_x \xrightarrow{p_{\xi_2}} c \), we know there exists \( c' \in \text{TCO} \) such that \( (w,c') \in G'.po_{\xi_2}r_T \). As such, we have \((r,c') \in G'.(r_{\xi_2}r_T;p_{\xi_2}r_T)\). From the definition of TCO we thus have \((\xi_1,\xi_2) \in \text{TCO} \). In case (2.b), from the definition of \( G.r_f \) we know there exists \( c' \in \text{TCO} \) such that \( (w,c') \in G'.r_f T \). As such, we have \((r,c) \in G'.(r_{\xi_2}r_T;r_f T)\). From the definition of TCO we thus have \((\xi_1,\xi_2) \in \text{TCO} \). In case (2.c) from the construction of \( \text{lo} \) we know there exists \( z \) such that either \( b = \xi_2,pl_x \) or \( b = \xi_2,ru_x \), and that there exist \( p,q \) such that \( \xi_2 \in S^p, \xi_2 \in S^q \) and \( p < q \). Given the definition of \( S^p \) we then know that there exists \( w \in G'.(\text{lo}) \), where either i) there exists \( w' \in \text{TCO} \) such that \( (w_z,w_z') \in G'.mo_{\xi_2 \xi_2} \); or ii) there exists \( r_z \in \text{TCO} \).
We proceed by contradiction. Let us assume that there exists \(a, \theta \) such that \((w, w') \in G'.(\text{mot}; \text{rf}_T)\). That is, we have either \((w, w') \in G'.\text{mot}\), or \((w, w') \in G'.\text{rf}_T\), or \((w, w') \in G'.(\text{mot}; \text{rf}_T)\). Moreover, since we have \((r, w) \in G'.\text{rb}_T\), we then have \((r, w') \in G'.(\text{rb}_T; \text{mot}; \text{rf}_T)\). From the definition of TCO we thus have \((\xi_1, \xi_c) \in \text{TCO}\).

Since \(c \xrightarrow{\text{hb}} a\), there are now two cases to consider: i) \(c \xrightarrow{\text{hb}} a\); or ii) \(c \xrightarrow{\text{hb}} a\).

In case (2.i), from Lemma 6 we have \(p_0 \cup (\xi_c, \xi_a) \in \text{TCO}\). As we have \((\xi_1, \xi_c) \in \text{TCO}\), we thus have \((\xi_1, \xi_a) \in \text{TCO}\), as required.

In case (2.ii) let us split the path from \(c\) at the first occurrence of a non-\(\text{hb}^\prime\) edge. That is, pick \(\xi_3, \xi_4, g, h, y, p, q, s, k\) such that \(c \xrightarrow{\text{hb}^\prime} \xi_3, g \xrightarrow{\text{lo}} \xi_4, h \xrightarrow{\text{hb}^\prime} a\), \(k < m\), \(P_s^m|_p = \xi_3\), \(P_s^m|_q = \xi_4\), \(p > q\) and either a) \(g = rl_g \wedge h = pl_y\); or b) \(g = ru_g \wedge h = pl_y\); or c) \(g = rl_g \wedge h = wu_y\); or d) \(g = ru_g \wedge h = wu_y\). From Lemma 6 we then have \((c, \xi_3, \xi_4, \xi_5) \in \text{TCO}\). As we have \((\xi_1, \xi_c) \in \text{TCO}\), we also have \((\xi_1, \xi_3) \in \text{TCO}\). We next demonstrate that in all cases (2.ii.a-2.ii.d) there exists \(t\) such that \(t \leq m\) and \(\xi_3, ru_y \xrightarrow{\text{lo}} \xi_4, pl_y \xrightarrow{\text{hb}^k} a\).

In case (2.ii.a) from the definition of \(\text{lo}\) we also have \(\xi_3, ru_y \xrightarrow{\text{lo}} \xi_4, pl_y \xrightarrow{\text{hb}^k} a\). As such, we have \(\xi_3, ru_y \xrightarrow{\text{lo}} \xi_4, pl_y \xrightarrow{\text{hb}^k} a\). In case (2.ii.b) the desired result holds immediately.

In cases (2.ii.c-2.ii.d) from the construction of \(\text{lo}\) we have \(\xi_3, ru_y \xrightarrow{\text{lo}} \xi_4, pl_y\). Moreover, since we have \(\xi_4, pl_y \xrightarrow{\text{po}} \xi_4, ru_y\), we also have \(\xi_4, pl_y \xrightarrow{\text{po}} \xi_4, ru_y \xrightarrow{\text{hb}^k} a\). As \(po \subseteq \text{hb}\) and \(\text{hb}\) is transitively closed, we have \(\xi_4, pl_y \xrightarrow{\text{hb}^k} a\). As such, we have \(\xi_3, ru_y \xrightarrow{\text{lo}} \xi_4, pl_y \xrightarrow{\text{hb}^k} a\). As \(k < m\), the desired result holds immediately.

Consequently, from (I.H.) we have \(\text{stg}(\xi_3, ru_y) \leq \text{stg}(a) \cup (\xi_3, \xi_a) \in \text{TCO}\). In the case of the first disjunct we have \(\text{stg}(\xi_3, ru_y) = \text{stg}(\xi_1, ru_y) \leq \text{stg}(a)\), as required. In the case of the second disjunct, since we also have \((\xi_1, \xi_3) \in \text{TCO}\) and TCO is transitively closed, we have \((\xi_1, \xi_a) \in \text{TCO}\), as required.

**Theorem 8 (Completeness).** For all SI execution graphs \(G'\) and their counterpoint implementation graphs \(G\) constructed as above,

\[\text{si-consistent}(G') \Rightarrow \text{RA-consistent}(G)\]

**Proof.** Pick an arbitrary SI execution graph \(G'\) and its counterpoint implementation graph \(G\) constructed as above and assume \(\text{si-consistent}(G')\) holds. From the definition of \(\text{RA-consistent}(G)\) it suffices to show:

1. \(\text{irreflexive}(G, \text{hb})\)
2. \(\text{irreflexive}(G, \text{mo}; G, \text{hb})\)
3. \(\text{irreflexive}(G, \text{rb}; G, \text{hb})\)

**RTS. part 1**

We proceed by contradiction. Let us assume that there exists \(a, \theta_e\) such that \(a \in \theta_e, E\) and \((a, a) \in G, \text{hb}\). There are now two cases to consider: 1) \((a, a) \in \text{hb}'\); or 2) \((a, a) \notin \text{hb}'\).
In case (1), from Lemma 6 we have \((a, a) \in G.\text{po}_1 \lor (T_\epsilon, T_\xi) \in \text{TCO}\). The first disjunct leads to a contradiction as the construction of \(G.\text{po}\) yields an acyclic relation. The second disjunct leads to a contradiction as \(\text{TCO}\) is a strict total order.

In case (2), let us split the \(a \xrightarrow{\text{hb}} a\) at the first occurrence of a non-\(\text{hb}'\) edge. That is, pick \(\xi_1, \xi_2, x, i, j, k, g, h\) such that \(a \xrightarrow{\xi_1, g} \xi_2, h \xrightarrow{\xi_2, h} a, P_1^i \mid j = T_{\xi_1}\), \(P_1^j \mid k = T_{\xi_2}\) and \(j > k\). As we have \(\xi_1, g \xrightarrow{\text{lo}} \xi_2, h \xrightarrow{\text{hb}} a \xrightarrow{\text{hb}} \xi_1, g\), from Lemma 7 and the definition of \(\text{hb}\) we then have \(\text{stg}(\xi_1, g) < \xi_1, g \lor (\xi_1, \xi_1) \in \text{TCO}\), leading to a contradiction in both disjuncts (the second disjunct yields a contradiction as \(\text{TCO}\) is a strict total order).

RTS. part 2
We proceed by contradiction. Let us assume that there exists \(a, \theta_{\xi_a}, b, \theta_{\xi_a}\) such that \(a \in \theta_{\xi_a}, E, b \in \theta_{\xi_a}, E, (a, b) \in G.\text{hb}\) and \((b, a) \in G.\text{mo}\). Let \(\text{loc}(a) = \text{loc}(b) = x\) for some shared location \(x\). There are now two cases to consider: 1) \((b, a) \in G.\text{mo}_1\); or 2) \((b, a) \in G.\text{mo}_2\).

In case (1) we then have \((b, a) \in G'.\text{mo}_1 \subseteq G'.\text{po}_1\). That is, we have \((b, a) \in G.\text{po} \subseteq G.\text{hb}\). We thus have \(a \xrightarrow{G.\text{hb}} b \xrightarrow{G.\text{hb}} a\), contradicting our proof in part 1. In case (2), from the construction of \(G.\text{mo}\) we have \((b, a) \in G'.\text{mo}_2\) and thus from the construction of \(G.\text{lo}\) we then have \((\xi_b, w_{ux}, \xi_a, r_{lx}) \in G.\text{lo}\). As such we have \(a \xrightarrow{G.\text{hb}} b \xrightarrow{G.\text{po}} \xi_b, w_{ux} \xrightarrow{G.\text{lo}} \xi_a, r_{lx} \xrightarrow{G.\text{po}} a\). That is, we have \(a \xrightarrow{G.\text{hb}} a\), contradicting our proof in part 1.

RTS. part 3
We proceed by contradiction. Let us assume that there exists \(a, \theta_{\xi_a}, b, \theta_{\xi_a}\) such that \(a \in \theta_{\xi_a}, E, b \in \theta_{\xi_a}, E, (a, b) \in G.\text{hb}\) and \((b, a) \in G.\text{rb}\).

Let \(\text{loc}(a) = \text{loc}(b) = x\) for some shared location \(x\). There are now two cases to consider: 1) \((b, a) \in G.\text{rb}_1\); or 2) \((b, a) \in G.\text{rb}_2\).

In case (1) we then have \((b, a) \in G'.\text{rb}_1 \subseteq G'.\text{po}_1\). That is, we have \((b, a) \in G.\text{po} \subseteq G.\text{hb}\). We thus have \(a \xrightarrow{G.\text{hb}} b \xrightarrow{G.\text{hb}} a\), contradicting our proof in part 1. In case (2), from the construction of \(G.\text{rb}\) we have \((b, a) \in G'.\text{rb}_2\) and thus from the construction of \(G.\text{lo}\) we then have either \((\xi_b, w_{ux}, \xi_a, r_{lx}) \in G.\text{po}\) and \((\xi_b, w_{ux}, \xi_a, r_{lx}) \in G.\text{lo}\); or \((b, \xi_a, r_{lx}) \in \text{po}\) and \((\xi_b, w_{ux}, \xi_a, r_{lx}) \in G.\text{lo}\). As such in both cases we have \((b, a) \in G.\text{hb}\). Consequently, we have \(a \xrightarrow{G.\text{hb}} b \xrightarrow{G.\text{hb}} a\), contradicting our proof in part 1.
D Soundness and Completeness of the Lazy SI Implementation

Given an execution graph $G$ of the lazy SI implementation, let us assign a transaction identifier to each transaction executed by the program; and given a transaction $\xi$. Let $RS^0 = WS^0 = \emptyset$. Observe that given a transaction $\xi$ of the lazy SI implementation, the trace of $\xi$, written $\theta_\xi$, is of the form:

$$F_s \xrightarrow{po|imm} I_s \xrightarrow{po|imm} T_s \xrightarrow{po|imm} RUs \xrightarrow{po|imm} PLs \xrightarrow{po|imm} Ws \xrightarrow{po|imm} Us$$

where:

- $F_s$ denotes the sequence of events failing to obtain the necessary locks, i.e. those iterations that do not succeed in promoting the writer locks;
- $I_s$ denotes the sequence of events initialising the values of $LS, RS$ and $WS$ with $\emptyset$, and initialising $s[x]$ with $\bot$ for each location $x$;
- $T_s$ denotes the sequence of events corresponding to the execution of $\langle T \rangle$ and is of the form $t_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t_k$, where for all $m \in \{1 \cdots k\}$:

$$t_m = \begin{cases} 
rd(x_m, v_m, RS_{m-1}, WS_{m-1}) \xrightarrow{po|imm} lr_{x_m} & \text{if } O_m = R(x_m, v_m) \\
wr(x_m, v_m, RS_{m-1}, WS_{m-1}) \xrightarrow{po|imm} ws_{x_m} \xrightarrow{po|imm} lw_{x_m} & \text{if } O_m = W(x_m, v_m)
\end{cases}$$

where $O_m$ denotes the $m$th event in the trace of the original $T$; $lr_{x_m} \triangleq R(s[x_m], v_m); lw_{x_m} \triangleq W(s[x_m], v_m); rs_{x_m} \triangleq R(x_m, v_m).$

$r_d(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} 
\emptyset & \text{otherwise}
\end{cases}$

$r_s_m$ denotes the sequence of events attempting (but failing) to acquire the read lock on $x_m, r_{l_{x_m}} \triangleq RL(x_1 m), wr_{r_{x_m}} \triangleq WR(RS, RS_m), rs_{x_m} \triangleq R(x_m, v_m),$ and $ws_{x_m} \triangleq W(s[x_m], v_m); and for all $m > 0$:

$$RS_{m+1} \triangleq \begin{cases} 
RS_m \cup \{x_m\} & \text{if } O_m = R(x_m, v_m) \\
RS_m & \text{otherwise}
\end{cases}$$

and

$$wr(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} 
\emptyset & \text{otherwise}
\end{cases}$$
Let \( W_S = W_{S_m} \) and \( WS = WS_m \); let \( RS \cup WS \) be enumerated as \( \{ x_1 \cdots x_n \} \) for some \( i \).

- \( RUs \) denotes the sequence of events releasing the reader locks (when the given location is in the read set only) and is of the form \( ru_{x_i} \rightarrow \cdots \rightarrow ru_{x_1} \), where for all \( n \in \{ 1 \cdots i \} \):
  \[
  ru_{x_n} = \begin{cases} 
  RU(x_n) & \text{if } x_n \notin WS_i \\
  \emptyset & \text{otherwise}
  \end{cases}
  \]

- \( PLs \) denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form \( pl_{x_1} \rightarrow \cdots \rightarrow pl_{x_i} \), where for all \( n \in \{ 1 \cdots i \} \):
  \[
  pl_{x_n} = \begin{cases} 
  PL(x_n) & \text{if } x_n \in WS_i \\
  \emptyset & \text{otherwise}
  \end{cases}
  \]

- \( Ws \) denotes the sequence of events committing the writes of \( LTM \) and is of the form \( w_{x_1} \rightarrow \cdots \rightarrow w_{x_i} \), where for all \( n \in \{ 1 \cdots i \} \):
  \[
  w_{x_n} = \begin{cases} 
  RU(x_n) \text{ if } x_n \notin WS_i \\
  WU(x_n) \text{ if } x_n \in WS_i \\
  \emptyset & \text{otherwise}
  \end{cases}
  \]

Given a transaction trace \( \xi \), we write e.g. \( \xi.Us \) to refer to its constituent \( Us \) sub-trace and write \( Us.E \) for the set of events related by \( po \) in \( Us \). Similarly, we write \( \xi.E \) for the set of events related by \( po \) in \( \xi.Us \). Note that \( G.E = \bigcup_{\xi \in Tx} \xi.E \).

Note that for each transaction \( \xi \) and each location \( x \), the \( \xi.rl_x, \xi.rs_x, \xi.ru_x, \xi.pl_x, \xi.wu_x \) and \( \xi.w_x \) are uniquely identified when they exist.

For each location \( x \in WS \), let \( fw_x \) denote the maximal write (in \( po \) order within \( \xi \)) logging a write for \( x \) in \( s[x] \). That is, when \( \theta_\xi = t_1 \rightarrow \cdots \rightarrow t_m \), let \( fw_x = \max(x, [t_1 \cdots t_m]) \), where

\[
\max(x, []) = \begin{cases} 
  lw_x & \text{undefined} \\
  lw_x & \text{if } t=wr(x, -, -) \rightarrow lw_x \rightarrow wws_x \\
  \max(x, L) & \text{otherwise}
  \end{cases}
\]
Similarly, for each location \( x \in WS_\xi \), let \( iw_x \) denote the minimal write (in \( po \) order within \( \xi \)) logging a write for \( x \) in \( s[x] \). That is, when \( \theta_\xi = t_1^{|po}|_{imm} \rightarrow \cdots \rightarrow t_m^{|po}|_{imm} \), let \( iw_x = \text{wmin}(x, [t_1 \cdots t_m]) \), where

\[
\text{wmin}(x,[]) \text{ undefined}
\]

\[
\begin{align*}
\text{wmin}(x,[t]+L) = \begin{cases} 
  lw_x & \text{if } t=wr(x,-,-) \xrightarrow{po} lw_x \xrightarrow{po} wws_x \\
  \text{wmin}(x,L) & \text{otherwise}
\end{cases}
\end{align*}
\]

D.1 Implementation Soundness

In order to establish the soundness of our implementation, it suffices to show that given an RA-consistent execution graph of the implementation \( G \), we can construct a corresponding SI-consistent execution graph \( G' \) with the same outcome.

Given a transaction \( \xi \in Tx \) with \( RS_\xi \cup WS_\xi = \{x_1 \cdots x_i\} \) and trace \( \theta_\xi \) as above with \( Ts = t_1^{|po}|_{imm} \rightarrow \cdots \rightarrow t_k \), we construct the corresponding SI execution trace \( \theta_\xi' \) as follows:

\[
\theta_\xi' = t_1^{|po}|_{imm} \rightarrow \cdots \rightarrow t_k'\]

where for all \( m \in \{1 \cdots k\} \):

\[
t_m' = R(x_m, v_m) \text{ when } t_m = rd(x_m, v_m, S_m) \xrightarrow{|po}|_{imm} lr_x_m
\]

\[
t_m' = \tilde{w}(x_m, v_m) \text{ when } t_m = wr(x_m, v_m, S_m) \xrightarrow{|po}|_{imm} lw_x_m \xrightarrow{|po}|_{imm} wws_x_m
\]

such that in the first case the identifier of \( t_m' \) is that of \( lr_x_m \); and in the second case the identifier of \( t_m' \) is that of \( lw_x_m \). We then define:

\[
RF_\xi = \begin{cases} 
  t_j' \in \theta_\xi'.E \land \exists x, v. t_j' = R(x, v) \land w = \tilde{w}(x, v) \\
  \land (w \in \xi, E \Rightarrow w \xrightarrow{|po}|_{imm} t_j' \land \\
  (\forall e \in \xi. E. w \xrightarrow{|po}|_{imm} e \xrightarrow{|po}|_{imm} t_j' \Rightarrow (loc(e) \neq x \lor e \notin W))) \\
  \land (w \notin \xi, E \Rightarrow (\forall e \in \xi. E. (e \xrightarrow{po} t_j' \Rightarrow (loc(e) \neq x \lor e \notin W))) \\
  \land \exists \xi'. (\xi', w_x, \xi, rs_x) \in G.rf) \land w = \xi'.f_w_x
\end{cases}
\]

Similarly, we define:

\[
MO = \begin{cases} 
  (w_1, w_2), \exists \xi, w_1, w_2 \in \xi, E \land W \land loc(w_1) = loc(w_2) \land (w_1, w_2) \in G.po \}
\end{cases}
\]

We are now in a position to demonstrate the soundness of our implementation. Given an RA-consistent execution graph \( G \) of the implementation, we construct an SI execution graph \( G' \) as follows and demonstrate that \( \text{si-consistent}(G') \) holds.

\[
- G'.E = \bigcup_{\xi \in Tx} \theta_\xi'.E, \text{ with the } \text{tx}(.) \text{ function defined as:}
\]

\[
\text{tx}(a) \triangleq \xi \text{ where } a \in \theta_\xi'
\]
That is, we have \( ξ \subseteq G \). The desired result follows immediately from the proof of part (20). In case (iii) the desired result holds trivially.

Assume \( (a, b) \in G' . ξ \). From the definition of (20) we then know \( (ξ_a, w_x, ξ_b, r_s) \in G . ξ \). On the other hand, from Lemma 2 we know that either i) \( x \in WS_x \) and \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \); or ii) \( x \notin WS_x \) and \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \); or iii) \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \). In case (i) we then have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \rightarrow G . ξ \) \( ξ_a, w_x \). That is, we have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_a, w_x \), contradicting the assumption that \( G \) is RA-consistent. Similarly in case (ii) we have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \rightarrow G . ξ \) \( ξ_a, w_x \). That is, we have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_a, w_x \), contradicting the assumption that \( G \) is RA-consistent. In case (iii) the desired result holds trivially.

Assume \( (a, b) \in G' . ξ \). From the definition of (20) we then know \( (ξ_a, w_x, ξ_b, w_x) \in G . ξ \). On the other hand, from Lemma 2 we know that either i) \( ξ_b, w_x \rightarrow G . ξ \) \( ξ_a, r_s \); or ii) \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, r_s \). In case (i) we then have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_b, w_x \rightarrow G . ξ \) \( ξ_a, r_s \). That is, we have \( ξ_a, w_x \rightarrow G . ξ \) \( ξ_a, w_x \), contradicting the assumption that \( G \) is RA-consistent. In case (ii) the desired result holds trivially.

Assume \( (a, b) \in G' . ξ \). From the definition of G' . ξ we then know that either \( (ξ_a, w_x, ξ_b, w_x) \in G . ξ \) or \( (ξ_a, r_s, ξ_b, w_x) \in G . ξ \). In the former case the desired result follows immediately from the proof of part (20). In the latter
We proceed by induction on case, from Lemma 2 we know that either i) \( \xi_a.wu_x \xrightarrow{G,bb} \xi_a.rl_x \); or ii) \( x \notin WS_\xi \) and \( \xi_a.ru_x \xrightarrow{G,bb} \xi_a.pl_x \); or iii) \( x \in WS_\xi \) and \( \xi_a.wu_x \xrightarrow{G,bb} \xi_b.rl_x \). In case (i) we then have \( \xi_b.wu_x \xrightarrow{G,bb} \xi_a.wu_x \xrightarrow{G,bb} \xi_a.rl_x \xrightarrow{G,po} \xi_a.rs_x \xrightarrow{G,bb} \xi_b.wu_x \). That is, we have \( \xi_b.wu_x \xrightarrow{G,bb} \xi_a.rs_x \xrightarrow{G,bb} \xi_b.wu_x \), contradicting the assumption that \( G \) is RA-consistent. In cases (ii-iii) the desired result holds trivially.

**Lemma 9.** For all RA-consistent execution graphs \( G \) of the implementation and their counterpart SI execution graphs \( G' \) constructed as above:

\[
\forall \xi_a, \xi_b. \forall a \in G'.T_{\xi_a}, b \in G'.T_{\xi_b}.
\]

\[
\forall a \in G'.T_{\xi_a}, b \in G'.T_{\xi_b}.
\]

\[
(a, b) \in G'.(po_T \cup rf_T \cup mo_T)^+ \Rightarrow \exists d \in G'.T_{\xi_a}, \forall c \in G'.T_{\xi_b}, (c, d) \in G.hb
\]

\[
\land (\exists y. (\xi_a.wu_y, d) \in G.hb \lor \forall c \in \xi_a.E. (c, d) \in G.hb)
\]

**Proof.** Let \( S_0 = G'.(po_T \cup rf_T \cup mo_T) \), and \( S_{n+1} = S_0; S_n \), for all \( n \geq 0 \). It is straightforward to demonstrate that \( G'.(po_T \cup rf_T \cup mo_T)^+ = \bigcup_{i \in N} S_i \). We thus demonstrate instead that:

\[
\forall i \in N. \forall \xi_a, \xi_b. \forall a \in G'.T_{\xi_a}, b \in G'.T_{\xi_b}.
\]

\[
(a, b) \in S_i \Rightarrow \exists d \in G'.T_{\xi_a}, \forall c \in G'.T_{\xi_b}, (c, d) \in G.hb
\]

\[
\land (\exists y. (\xi_a.wu_y, d) \in G.hb \lor \forall c \in \xi_a.E. (c, d) \in G.hb)
\]

We proceed by induction on \( i \).

**Base case** \( i = 0 \)

Pick arbitrary \( \xi_a, \xi_b, a \in G'.T_{\xi_a}, b \in G'.T_{\xi_b} \) such that \( (a, b) \in S_0 \). There are now three cases to consider: 1) \( (a, b) \in G'.po_T \); or 2) \( (a, b) \in G'.rf_T \); or 3) \( (a, b) \in G'.mo_T \).

In case (1), pick an arbitrary \( c \in G'.T_{\xi_a} \). From the definition of \( G'.po_T \) we have \( (c, b) \in G'.po \subseteq G.hb \), as required. Pick an arbitrary \( c \in \xi_a.E. \). From the definition of \( G'.po_T \) we have \( (c, b) \in G'.po \subseteq G.hb \), as required.

In case (2), we then know there exists \( w \in G'.T_{\xi_a} \) and \( r \in G'.T_{\xi_b} \) such that \( (w, r) \in G'.rf \). Let \( loc(w) = loc(r) = x \). From Lemma 8 we then have \( \xi_a.wu_x \xrightarrow{G,bb} \xi_b.rl_x \). Pick an arbitrary \( c \in G'.T_{\xi_a} \). As such we have \( c \xrightarrow{G,po} \xi_a.wu_x \xrightarrow{G,bb} \xi_b.rl_x \xrightarrow{G,po} \xi_b.rs_x \). That is, we have \( (c, \xi_b.rs_x) \in G.hb \), as required.

In case (3), we then know there exists \( w \in G'.T_{\xi_a} \) and \( w' \in G'.T_{\xi_b} \) such that \( (w, w') \in G'.mo \). Let \( loc(w) = loc(w') = x \). From Lemma 8 we then have \( \xi_a.wu_x \xrightarrow{G,bb} \xi_b.rl_x \). Pick an arbitrary \( c \in G'.T_{\xi_a} \). As we also have \( \xi_b.rl_x \xrightarrow{G,po} w' \) and \( c \xrightarrow{G,po} \xi_a.wu_x \), we then have \( (c, w'), (\xi_a.wu_x, w') \in G.hb \), as required.

**Inductive case** \( i = n+1 \)

\[
\forall j \in N. \forall \xi_a, \xi_b, \forall a \in G'.T_{\xi_a}, b \in G'.T_{\xi_b}.
\]

\[
(a, b) \in S_j \land j \leq n \Rightarrow \exists d \in G'.T_{\xi_a}, \forall c \in G'.T_{\xi_b}, (c, d) \in G.hb
\]

\[
\land (\exists y. (\xi_a.wu_y, d) \in G.hb \lor \forall c \in \xi_a.E. (c, d) \in G.hb)
\] (I.H.)
We proceed by contradiction. Let us assume $G$ is RA-consistent. From the definition of $S_i$ we then know there exist $e, \xi_e$ such that $e \in G'.T_{\xi_e}$, $(a, e) \in S_0$ and $(e, b) \in S_n$. Since $(e, b) \in S_n$, from (I.H.) we know there exists $d \in G'.T_{\xi_b}$ such that $\forall f' \in G'.T_{\xi_e}, (f', d) \in G.\text{hb}$. On the other hand, from the proof of the base case we know there exists $f \in G'.T_{\xi_e}$ such that $\forall e' \in G'.T_{\xi_e}, (e', f) \in G.\text{hh}$; and that $\exists y. (\xi_e.wu_y, f) \in G.\text{hb} \lor \forall e' \in \xi_a.E. (e', f) \in G.\text{hb}$. Pick an arbitrary $c \in G'.T_{\xi_c}$. We thus know that $(c, f) \in G.\text{hb}$. As $f \in G'.T_{\xi_e}$, we thus have $(f, d) \in G.\text{hb}$ if $G.\text{hb}$ is transitively closed and we have $(c, f), (d, e) \in G.\text{hb}$ and $\exists y. (\xi_e.wu_y, f) \in G.\text{hb} \lor \forall e' \in \xi_a.E. (e', f) \in G.\text{hb}$, we then have $(c, d) \in G.\text{hb}$, and $\exists y. (\xi_e.wu_y, d) \in G.\text{hb} \lor \forall e' \in \xi_a.E. (e', d) \in G.\text{hb}$, as required.

Theorem 9 (Soundness). For all execution graphs $G$ of the implementation and their counterpart SI execution graphs $G'$ constructed as above, $G$ is RA-consistent.

Proof. Pick an arbitrary execution graph $G$ of the implementation such that $G$ is RA-consistent, and its associated SI execution graph $G'$ constructed as described above.

RTS. irreflexive(si-hb)

We proceed by contradiction. Let us assume $\neg$irreflexive($G'.(po_T \cup rf_T \cup mo_T \cup si-rb)$). Let $S = G'.(po_T \cup rf_T \cup mo_T \cup si-rb)$. There are two cases to consider: either there is a cycle with no si-rb edges; or there is a cycle with one or more si-rb edges. That is, either 1) there exists a such that $(a, a) \in S$; or 2) there exist $a_1, b_1, \cdots, a_n, b_n$ such that $a_1 \xrightarrow{G'.si-rb} b_1 \xrightarrow{S} a_2 \xrightarrow{G'.si-rb} b_2 \xrightarrow{S} \cdots \xrightarrow{S} a_n \xrightarrow{G'.si-rb} b_n \xrightarrow{S} a_1$.

In case (1) we then know there exists $\xi$ such that $a \in G'.T_{\xi_a}$. As such, from Lemma 9 we know that there exists $d \in G'.T_{\xi_a}$ such that for all $c \in G'.T_{\xi_c}$, $(c, d) \in G.\text{hb}$. As such, we have $(d, d) \in G.\text{hb}$, contradicting the assumption that $G$ is RA-consistent.

In case (2), for an arbitrary $i \in \{1 \cdots n\}$, let $j = i + 1$ when $i \neq n$; and $j = 1$ when $i = n$. As $a_i \xrightarrow{G'.si-rb} b_i$, we know there exists $\xi_a, \xi_b$ such that $a_i \in T_{\xi_a}$, $b_i \in T_{\xi_b}$, and that there exist $r_i \in T_{\xi_a} \cap R_E$ and $w_i \in T_{\xi_b} \cap W$ such that $(r_i, w_i) \in G'.rb$. Let $\text{loc}(r_i) = \text{loc}(w_i) = x_i$. From Lemma 8 we then know that either i) $\xi_a.ru_{x_i} \xrightarrow{G.\text{hb}} \xi_b.pl_{x_i}$; or ii) $\xi_a.wu_{x_i} \xrightarrow{G.\text{hb}} \xi_b.rl_{x_i}$. Note that for all $y$ such that $\xi_a.wu_y$ exists, we know $\xi_b.rl_{x_i} \xrightarrow{G.\text{po}} \xi_b.wu_y$ and $\xi_b.pl_{x_i} \xrightarrow{G.\text{po}} \xi_b.wu_y$. As such, as $(b_i, a_j) \in S$, from Lemma 9 and since $G.\text{hb}$ is transitively closed, we know there exists $d_j \in T_{\xi_{a_j}}$ such that either $\xi_a.ru_{x_i} \xrightarrow{G.\text{hb}} d_j$, or $\xi_a.wu_{x_i} \xrightarrow{G.\text{hb}} d_j$.

That is, $\xi_a.ru_{x_i} \xrightarrow{G.\text{hb}} d_j$, where either $u_{x_i} = r_{x_i}$, or $u_{x_i} = w_{x_i}$. On the other hand, observe that for all $d_i \in T_{\xi_{a_i}}$, we have $d_i \xrightarrow{G.\text{po}} \xi_{a_i}u_{x_i}$ i.e. $d_i \xrightarrow{G.\text{hb}} \xi_{a_i}u_{x_i}$. As such, we have $d_j \xrightarrow{G.\text{hb}} \xi_{a_j}u_{x_j}$. As we also have $\xi_{a_i}u_{x_i} \xrightarrow{G.\text{hb}} d_j$ and $G.\text{hb}$ is transitively closed, we have $\xi_{a_i}u_{x_i} \xrightarrow{G.\text{hb}} \xi_{a_j}u_{x_j}$. We then have $\xi_{a_1}u_{x_1} \xrightarrow{G.\text{hb}} \xi_{a_2}u_{x_2} \xrightarrow{G.\text{hb}} \cdots \xrightarrow{G.\text{hb}} \xi_{a_n}u_{x_n} \xrightarrow{G.\text{hb}} \xi_{a_1}u_{x_1}$.
that $G$ is RA-consistent.

**RTS.** $rf_1 \cup mo_1 \cup rb_1 \subseteq po$

Follows immediately from the construction of $G'$.

### D.2 Implementation Completeness

In order to establish the completeness of our implementation, it suffices to show that given an SI-consistent execution graph $G' = (E, po, rf, mo, lo)$, we can construct a corresponding RA-consistent execution graph $G$ of the implementation.

Note that the execution trace for each transaction $T \in G'.T / st$ is of the form $\theta'_k = \tau'_1 po^{l_{\text{imm}}} \cdots \tau'_k po^{l_{\text{imm}}} \tau'_1$, for some $k$, where each $\tau'_i$ is a read or write event. As such, we have $G'.E = G'.T = \bigcup_{T \in G'.T / st} \theta'_k = \theta'_k E$. We thus construct the implementation trace $\theta_k$ as follows:

$$
I_s po^{l_{\text{imm}}} T_s po^{l_{\text{imm}}} RUs po^{l_{\text{imm}}} PLS po^{l_{\text{imm}}} Ws po^{l_{\text{imm}}} Us
$$

where:

- $I_s$ denotes the sequence of events initialising the values of $RS$ and $WS$ with $\emptyset$, and initialising $s[x]$ with $\bot$ for each location $x$;
- $T_s$ is of the form $t_1 po^{l_{\text{imm}}} \cdots \tau'_k po^{l_{\text{imm}}} \tau'_k$, where for all $m \in \{1 \cdots k\}$:

$$
t_m = \begin{cases} 
rd(x_m, v_m, RS_{m-1}, WS_{m-1}) po^{l_{\text{imm}}} lw_{x_m} & \text{if } \tau'_m = R(x_m, v_m) \\
wr(x_m, v_m, RS_{m-1}, WS_{m-1}) po^{l_{\text{imm}}} lw_{x_m} & \text{if } \tau'_m = W(x_m, v_m)
\end{cases}
$$

where $lw_{x_m} = R(s[x_m], v_m)$; the identifiers of $lw_{x_m}$ are those of $\tau'_m$, whilst the identifiers of other events in $t_m$ are picked fresh;

$$
rd(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} 
R(s[x_m], \bot) & \text{if } x_m \notin RS_{m-1} \cup WS_{m-1} \\
po^{l_{\text{imm}}} lw_{x_m} & \text{if } \tau'_m = R(x_m, v_m) \\
po^{l_{\text{imm}}} lw_{x_m} & \text{if } \tau'_m = W(x_m, v_m)
\end{cases}
$$

$ws_{x_m} \triangleq w(RS, RS_m)$, $fs_{x_m}$ denotes the sequence of events attempting (but failing) to acquire the read lock on $x_m$, $r_{x_m} \triangleq RL(x1_m)$, $rs_{x_m} \triangleq R(x_m, v_m)$, and $ws_{x_m} \triangleq w(s[x_m], v_m)$; $RS_0 = \emptyset$ and for all $m > 0$:

$$
RS_{m+1} \triangleq \begin{cases} 
RS_m \cup \{x_m\} & \text{if } \tau'_m = R(x_m, -) \\
RS_m & \text{otherwise}
\end{cases}
$$
and

\[ \text{wr}(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} p_0 |_{\text{imm}} \rightarrow f_{x_m} & \text{if } x_m \not\in RS_{m-1} \cup WS_{m-1} \\ p_0 |_{\text{imm}} \rightarrow r_{x_m} \emptyset & \text{otherwise} \end{cases} \] 

\[ lw_{x_m} = \overline{w(s[x_m], v_m)}, \quad wws_{x_m} \triangleq w(WS, WS_m); \]  
\[ f_{x_m} \text{ and } rl_{x_m} \text{ are as defined above; and} \]

\[ WS_{m+1} \triangleq \begin{cases} WS_m \cup \{x_m\} & \text{if } t'_m=\overline{w(x_m, -)} \\ WS_m & \text{otherwise} \end{cases} \]

Let \( RS_\xi = RS_m \) and \( WS_\xi = WS_m \); let \( RS_\xi \cup WS_\xi \) be enumerated as \( \{x_1, \ldots, x_i\} \) for some \( i \).

- \( RU_\xi \) denotes the sequence of events releasing the reader locks (when the given location is in the read set only) and is of the form \( ru_{x_1} \overset{p_0 |_{\text{imm}}}{\rightarrow} \cdots \overset{p_0 |_{\text{imm}}}{\rightarrow} ru_{x_i} \) where for all \( n \in \{1, \ldots, i\} \):

\[ ru_{x_n} = \begin{cases} RU(x_{1,n}) & \text{if } x_n \not\in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]

with the identifier of each \( ru_{x_n} \) picked fresh;

- \( PL_\xi \) denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form \( pl_{x_1} \overset{p_0 |_{\text{imm}}}{\rightarrow} \cdots \overset{p_0 |_{\text{imm}}}{\rightarrow} pl_{x_i} \), where for all \( n \in \{1, \ldots, i\} \):

\[ pl_{x_n} = \begin{cases} PL(x_{1,n}) & \text{if } x_n \in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]

with the identifier of each \( ru_{x_n} \) picked fresh;

- \( W_\xi \) denotes the sequence of events committing the writes of \( (T) \) and is of the form \( c_{x_1} \overset{p_0 |_{\text{imm}}}{\rightarrow} \cdots \overset{p_0 |_{\text{imm}}}{\rightarrow} c_{x_i} \), where for all \( n \in \{1, \ldots, i\} \):

\[ c_{x_n} = \begin{cases} R(s[x_n], v_n) \overset{p_0 |_{\text{imm}}}{\rightarrow} w_{x_n} = \overline{w(x_n, v_n)} & \text{if } x_n \in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]

with the identifiers of events in each \( c_{x_n} \) picked fresh;

- \( Us_\xi \) denotes the sequence of events releasing the locks on the write set, and is of the form \( wu_{x_1} \overset{p_0 |_{\text{imm}}}{\rightarrow} \cdots \overset{p_0 |_{\text{imm}}}{\rightarrow} wu_{x_i} \), where for all \( n \in \{1, \ldots, i\} \):

\[ wu_{x_n} = \begin{cases} WU(x_{1,n}) & \text{if } x_n \in WS_\xi \\ \emptyset & \text{otherwise} \end{cases} \]

with the identifier of each \( wu_{x_n} \) picked fresh.
We use the $\xi$ prefix to project the various events of the implementation trace $\theta_\xi$ (e.g. $\xi.rlx_\xi$). Note that for each transaction $\xi$ and each location $x$, the $\xi.rlx_\xi$, $\xi.rsx_\xi$, $\xi.rux_\xi$, $\xi.plx_\xi$, $\xi.wux_\xi$ and $\xi.wx_\xi$ are uniquely identified when they exist.

For each location $x \in WS_\xi$, let $fw_x$ denote the maximal write (in po order within $\xi$) logging a write for $x$ in $s[x]$. That is, when $\theta_\xi = t_1 \to \cdots \to t_m$, let $fw_x = \max(x, [t_1 \cdots t_m])$, where

$$wmax(x, [t]) \triangleq \begin{cases} 
    lw_x & \text{if } t = wr(x, -, -, -) \to lw_x \to wws_x \\
    wmax(x, L) & \text{otherwise}
\end{cases}$$

Similarly, for each location $x \in WS_\xi$, let $iw_x$ denote the minimal write (in po order within $\xi$) logging a write for $x$ in $s[x]$. That is, when $\theta_\xi = t_1 \to \cdots \to t_m$, let $iw_x = \min(x, [t_1 \cdots t_m])$, where

$$wmin(x, [t]) \triangleq \begin{cases} 
    lw_x & \text{if } t = wr(x, -, -, -) \to lw_x \to wws_x \\
    wmin(x, L) & \text{otherwise}
\end{cases}$$

Analogously, for each location $x \in RS_\xi$, let $ir_x$ denote the minimal read (in po order within $\xi$) reading the value of $x$. That is, when $\theta_\xi = t_1 \to \cdots \to t_m$, let $ir_x = \min(x, [t_1 \cdots t_m])$, where

$$wmin(x, [t]) \triangleq \begin{cases} 
    lw_x & \text{if } t = rd(x, -, -, -) \to lw_x \to wws_x \\
    wmin(x, L) & \text{otherwise}
\end{cases}$$

Given the $G'$ classes $T/st$, let us construct the strict total order $TCO : T/st \times T/st$ as described in Appendix C.2. For each location $x$ and $i \in \mathbb{N}$, let us similarly define $S^x_i$, $P^x_i$, $P^x_i.writer$, $LO_x$, $RLO_x$, $lo_1$ and $lo_2$.

Remark 5. Recall that both $lo_1$ and $lo_2$ satisfy the conditions stated in Def. 4; $lo_2$ additionally satisfies the ‘read-read-synchronisation’ property in (RSync). As before, we demonstrate that given an SI-consistent execution graph $G'$, it is always possible to construct an RA-consistent execution graph $G$ of the implementation with its lock order defined as $lo_2$. Recall that as $lo_1 \subseteq lo_2$, it is straightforward to show that replacing $lo_2$ in such a $G$ with $lo_1$, preserves the RA-consistency of $G$, as defined in Def. 4. In other words, as $lo_1 \subseteq lo_2$, we have:

$$\text{acyclic}(hb_{loc} \cup mo \cup rb) \text{ with } hb \triangleq (po \cup rf \cup lo_2)^+ \Rightarrow \text{acyclic}(hb_{loc} \cup mo \cup rb) \text{ with } hb \triangleq (po \cup rf \cup lo_1)^+$$

As such, by establishing the completeness of our implementation with respect to $lo_2$, we also establish its completeness with respect to $lo_1$. In other words, we demonstrate the completeness of our implementation with respect to both lock implementations presented earlier in Appendix A.
We now demonstrate the completeness of our implementation. Given an SI-consistent graph, we construct an implementation graph $G$ as follows and demonstrate that it is RA-consistent.

- $G.E = \bigcup_{T \in G'.T/st} \theta_{\xi}.E$, with the $tx(e) = 0$, for all $e \in G.E$.
- $G.po$ is defined as $G'.po$ extended by the po for the additional events of $G$, given by each $\theta_{\xi}$ trace defined above. Note that $G.po$ does not introduce additional orderings between events of $G'.E$. That is, $\forall a,b \in G'.E. (a,b) \in G'.po \iff (a,b) \in G.po$.
- $G.rf = \bigcup_{x \in TX} RF_{\xi}$ with $RF_{\xi} \triangleq \{ (w,\xi.rs_x) \mid x \in Loc \land (w,\xi.ir_x) \in G'.rf \}$.
- $G.mo = \{ (\xi_1,w_1,\xi_2,w_2) \mid \xi_1,\xi_2, x \in TX \land x \in Loc \land (\xi_1-fw_x,\xi_2.iw_x) \in G'.mo^+ \}$.
- $G.lo = \bigcup_{x \in LOC} RLO_x$, with $RLO_x$ as defined above.

**Notation** Given an implementation graph $G$ as constructed above (with $G.T = \emptyset$), and a relation $r \subseteq G.E \times G.E$, we override the $r_E$ notation and write $r_E$ for:

$$\{ (a,b) \in r \mid \exists \xi_a, \xi_b. a \in \theta_{\xi_a}.E \land b \in \theta_{\xi_b}.E \land \xi_a \neq \xi_b \}$$

Analogously, we write $r_I$ for $\{ (a,b) \in r \mid \exists \xi. a, b \in \theta_{\xi}.E \}$.

**Lemma 10.** For all SI-consistent execution graphs $G' = (E,po,rf,mo,lo)$, and for all $x \in Loc$, $\xi_a,\xi_b \in TXIn$, and $i,j \in \mathbb{N}$:

$$\xi_a \in S_x^i \land \xi_b \in S_x^j \land i < j \Rightarrow (\xi_a,\xi_b) \in TCO$$

**Proof.** Follows immediately from Corollary 1.

Given an implementation graph $G = (E,po,rf,mo,lo)$, let us define $hb'$ and $hb''$ as described in Appendix C.2 for all $i \in \mathbb{N}$.

**Lemma 11.** For all implementation graphs $G = (E,po,rf,mo,lo)$ constructed as above,

$$\forall a, b, \xi_a, \xi_b. a \in \theta_{\xi_a}.E \land b \in \theta_{\xi_b}.E \land a \to b \Rightarrow (a,b) \in G.po \lor (\xi_a,\xi_b) \in TCO$$

**Proof.** Pick an arbitrary implementation graph $G = (E,po,rf,mo,lo)$ constructed as above. Since $hb'$ is a transitive closure, it is straightforward to demonstrate that $hb' = \bigcup_{i \in \mathbb{N}} hb''$. It thus suffices to show:

$$\forall a, b, \xi_a, \xi_b. a \in \theta_{\xi_a}.E \land b \in \theta_{\xi_b}.E \land a \to b \Rightarrow (a,b) \in G.po \lor (\xi_a,\xi_b) \in TCO$$

We proceed by induction over $i$. 
**Base case** \( i = 0 \)

Pick arbitrary \( a, b, \xi_a, \xi_b \) such that \( a \in \theta_{\xi_a}, E \) and \( b \in \theta_{\xi_b}, E \), \( a \xrightarrow{\text{hb}} b \). There are then four cases to consider: 1) \( a \xrightarrow{G, \text{po}_E} b \); or 2) \( a \xrightarrow{G, \text{rf}} b \); 3) \( a \xrightarrow{G, \text{lo}_E} b \); or 4) \( a \xrightarrow{G, \text{lo}_E} b \).

In case (1), from the construction of \( G, \text{rf}_E \) we have \( G, \text{rf}_E \subseteq G, \text{po}_E \); moreover, from the construction of \( G, \text{lo}_E \) we have \( G, \text{lo}_E \subseteq G, \text{po}_E \). Consequently we have \( a \xrightarrow{G, \text{po}_E} b \), as required.

In case (2) from the construction of \( G, \text{po} \) we then know that \( T_{\xi_a} \times T_{\xi_b} \subseteq G', \text{po} \) and thus from the definition of \( \text{TCO} \) we have \((\xi_a, \xi_b) \in \text{TCO}, \) as required.

In case (3) from the construction of \( G, \text{rf} \) we then know that \( T_{\xi_a} \times T_{\xi_b} \subseteq G', \text{rf}_T \) and thus from the definition of \( \text{TCO} \) we have \((\xi_a, \xi_b) \in \text{TCO}, \) as required.

In case (4) from the definition of \( \text{hb}_n \) we know that there exists \( i, j, i', j' \) such that \( \xi_a = P_{i'} \upharpoonright j \), \( \xi_b = P_i \upharpoonright j' \) and either \( i < i' \); or ii) \( i = i' \land j < j' \).

In case (4.i) from Lemma 10 we have \((\xi_a, \xi_b) \in \text{TCO}, \) as required. In case (4.ii) from the definitions of \( P_{i'} \) we have \((\xi_a, \xi_b) \in \text{TCO}, \) as required.

**Inductive case** \( i = n+1 \)

Pick arbitrary \( a, b, \xi_a, \xi_b \) such that \( a \in \theta_{\xi_a}, E \), \( b \in \theta_{\xi_b}, E \) and \( a \xrightarrow{\text{hb}} b \).

\[
\forall j \leq n. \forall c, d, \xi_c, \xi_d. c \in \theta_{\xi_c}, E \land d \in \theta_{\xi_d}, E \land c \xrightarrow{\text{hb}_n'} d \Rightarrow (c, d) \in G, \text{po}_E \lor (\xi_c, \xi_d) \in \text{TCO}
\] (I.H.)

From the definition of \( \text{hb}_{n+1} \) we then know there exists \( e \) such that \( (a, e) \in \text{hb}_n \) and \( (e, b) \in \text{hb}_n' \). Let \( e \in \xi_e, E \). Consequently, from the proof of the base case we then know that \( (a, e) \in G, \text{po}_E \lor (\xi_a, \xi_e) \in \text{TCO}. \) Similarly, from (I.H.) we have \( (e, b) \in G, \text{po}_E \lor (\xi_e, \xi_b) \in \text{TCO} \). There are now four cases to consider: 1) \( (a, e) \in G, \text{po}_E \) and \( (e, b) \in G, \text{po}_E \); 2) \( (a, e) \in G, \text{po}_E \) and \( (\xi_e, \xi_b) \in \text{TCO} \); 3) \((\xi_a, \xi_e) \in \text{TCO} \) and \( (e, b) \in G, \text{po}_E \); 4) \((\xi_e, \xi_b) \in \text{TCO} \) and \( (\xi_a, \xi_c) \in \text{TCO} \).

In case (1) from the definition of \( G, \text{po}_E \) we have \((a, b) \in G, \text{po}_E \). In cases (2-4) from the definitions of \( G, \text{po}_E \) we have \((\xi_a, \xi_b) \in \text{TCO} \), as required.

Given an implementation graph \( G = (E, \text{po}, \text{rf}, \text{mo}, \text{lo}) \) constructed as above, let \( \text{stg}(\cdot) : G, E \rightarrow \{1, 2, 3\} \) denote the *event stage* as follows:

\[
\text{stg}(e) \overset{\triangleq}{=} \begin{cases} 
1 & \exists \xi, e \in \xi.Ts \\
2 & \exists \xi, x. e = \xi.ru_x \\
3 & \text{otherwise}
\end{cases}
\]
Lemma 12. For all implementation graphs \( G = (E, po, rf, mo, lo) \) constructed as above, and for all \( x \in \text{Loc} \), \( \xi_0, \xi_1, \xi_2 \in \text{TXIn} \) and \( a \in \theta_{\xi_0}.E \):

\[
\left( \xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a \right) \land (\exists j, k, l. P^x_j \downarrow k, l. P^x_j \downarrow = \xi_1 \land P^x_j \downarrow = \xi_2 \land k > l) \Rightarrow \text{stg}(\xi_1.r_x) < \text{stg}(a) \lor (\xi_1, \xi_2) \in \text{TCO}
\]

Proof. Pick an arbitrary implementation graph \( G = (E, po, rf, mo, lo) \) constructed as above, and pick arbitrary \( x \in \text{Loc} \), \( \xi_0, \xi_1, \xi_2 \in \text{TXIn} \) and \( a \in \theta_{\xi_0}.E \) such that

\[
\left( (\xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a) \lor (\xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a) \lor (\xi_1.r_x \xrightarrow{lo} \\
\xi_2.wu_x \land \xi_2.wu_x \xrightarrow{hb} a) \lor (\xi_1.r_x \xrightarrow{lo} \\
\xi_2.wu_x \land \xi_2.wu_x \xrightarrow{hb} a) \right) \land (\exists j, k, l. P^x_j \downarrow = \xi_1 \land P^x_j \downarrow = \xi_2 \land k > l).
\]

In case of the first disjunct from the construction of \( lo \) we then also have \( \xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a \). Similarly, in case of the third and fourth disjuncts from the construction of \( lo \) we also have \( \xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.wu_x \xrightarrow{hb} a \). Moreover, since we have \( \xi_2.pl_x \xrightarrow{po} \xi_2.wu_x \) and \( po \subseteq hb \), we also have \( \xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a \). It thus suffices to show:

\[
\xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a \land (\exists j, k, l. P^x_j \downarrow = \xi_1 \land P^x_j \downarrow = \xi_2 \land k > l) \Rightarrow \text{stg}(\xi_1.r_x) < \text{stg}(a) \lor (\xi_1, \xi_2) \in \text{TCO}
\]

Since \( G.hb \) is a transitive closure, it is straightforward to demonstrate that \( G.hb = \bigcup_{i \in \mathbb{N}} \text{hb}_i \), where \( \text{hb}_0 = G.po \cup G.rf \cup G.lo \) and \( \text{hb}_{i+1} = \text{hb}_i \cup \text{hb}_i \). It thus suffices to show:

\[
\forall i \in \mathbb{N}. \forall x \in \text{Loc}, \xi_0, \xi_1, \xi_2 \in \text{TXIn}, a \in \theta_{\xi_0}.E. \\
\xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \land \xi_2.pl_x \xrightarrow{hb} a \land (\exists j, k, l. P^x_j \downarrow = \xi_1 \land P^x_j \downarrow = \xi_2 \land k > l) \Rightarrow \text{stg}(\xi_1.r_x) < \text{stg}(a) \lor (\xi_1, \xi_2) \in \text{TCO}
\]

We thus proceed by induction over \( i \).

**Base case** \( i = 0 \)

Pick arbitrary \( x \in \text{Loc}, \xi_0, \xi_1, \xi_2 \in \text{TXIn} \) and \( a \in \theta_{\xi_0}.E \) such that \( \xi_1.r_x \xrightarrow{lo} \xi_2.pl_x \), \( \xi_2.pl_x \xrightarrow{hb} a \) and \( \exists j, k, l. P^x_j \downarrow = \xi_1 \land P^x_j \downarrow = \xi_2 \land k > l \). From the construction of \( G \) we know that there exists \( r \in G'.\mathcal{T}_{\xi_1} \) and \( w \in G'.\mathcal{T}_{\xi_2} \) such that \( (r, w) \in G'.rb \). Moreover, since \( k \neq l \) we know that \( (r, w) \in G'.rb \). Since we have \( \xi_2.pl_x \xrightarrow{hb} a \), there are four cases to consider: 1) \( \xi_2.pl_x \xrightarrow{G.po} a \); or 2) \( \xi_2.pl_x \xrightarrow{G.p_of} a \); or 3) \( \xi_2.pl_x \xrightarrow{G.rf} a \); or 4) \( \xi_2.pl_x \xrightarrow{G.lo} a \).

In case (1), from the construction of \( G.rf \) we have \( G.rf \subseteq G.po \); moreover, from the construction of \( G.lo \) we have \( G.lo \subseteq G.po \). We thus have \( \xi_2.pl_x \xrightarrow{G.po} a \).
On the other hand, from the definition of \( \text{stg}(\cdot) \) we know \( \text{stg}(a) \geq \text{stg}(\xi_2.pl_x) > \text{stg}(\xi_1.ru_x) \), and thus \( \text{stg}(a) > \text{stg}(\xi_1.ru_x) \), as required.

In case (2), since \( \xi_2.pl_x \xrightarrow{\text{po}} a \), we know there exists \( a' \in T_{\xi_a} \) such that \( (w, a') \in G'.po_T \). As such, we have \( (r, a') \in G'.(rb_T; po_T) \). From the definition of TCO we thus have \( (\xi_1, \xi_a) \in \text{TCO} \), as required.

Case (3) cannot happen as there are no \( \text{rf} \) edges from lock events. In case (4), from the construction of \( \text{lo} \) we know there exists \( m, n \) such that \( m > j \), and \( \xi_a = P^x_{m,n} \). As such, from Lemma 10 we have \( (\xi_1, \xi_a) \in \text{TCO} \), as required.

**Inductive case** \( i = n+1 \)

Pick arbitrary \( x \in \text{Loc} \) and \( \xi_a, \xi_1, \xi_2 \in \text{TXI}d \) and \( a \in \theta_{\xi_a}.E \) such that \( \xi_1.ru_x \xrightarrow{\text{lo}} \xi_2.pl_x, \xi_2.pl_x \xrightarrow{\text{hb}} a \) and \( \exists j, k, l. \ P^x_j \mid_k = \xi_1 \land P^x_j \mid_l = \xi_2 \land k > l. \)

\[ \forall j \leq n. \forall y \in \text{Loc}, \xi_a', \xi_3, \xi_4 \in \text{TXI}d, a' \in \theta_{\xi_a}.E. \]

\[ (\xi_3.ru_x \xrightarrow{\text{lo}} \xi_4.pl_x \land \xi_4.pl_x \xrightarrow{\text{hb}} a') \land \exists j, k, l. \ P^x_j \mid_k = \xi_1 \land P^x_j \mid_l = \xi_2 \land k > l \]

\[ \Rightarrow \text{stg}(\xi_3.ru_x) < \text{stg}(a') \lor (\xi_3, \xi_a') \in \text{TCO} \]

(I.H.)

There are two cases to consider:

1) \( \xi_2.pl_x \xrightarrow{(po \cup rf)(\text{Ulo})} a \); or

2) \( \xi_2.pl_x \xrightarrow{(po \cup rf)(\text{Ulo})} (pl \cup \text{Ulo});(po \cup Urf)(\text{Ulo}) \xrightarrow{\text{hb}} a \), where \( m \leq n \).

In case (1), from the construction of \( G.rf_1 \) we have \( G.rf \subseteq G.po_1 \); moreover, from the construction of \( G.lo_1 \) we have \( G.lo_1 \subseteq G.po_1 \). We thus have \( \xi_2.pl_x \xrightarrow{po} a \), i.e. \( \xi_2.pl_x \xrightarrow{po} a \). As such, from the proof of the base case we have \( \text{stg}(a) > \text{stg}(\xi_1.ru_x) \), as required.

In case (2) from the construction of \( G \) we know that there exists \( r \in \theta_{\xi_a}.E \) and \( w \in \theta_{\xi_2}.E \) such that \( (r, w) \in G'.rb \). Moreover, since \( k \neq l \), from the construction of \( G \) we know that \( \xi_1 \neq \xi_2 \) and thus we have \( (r, w) \in G'.rb_T \). On the other hand, we know that there exist \( b, c \) such that \( \xi_2.pl_x \xrightarrow{(po \cup Urf)(\text{Ulo})} b \xrightarrow{\text{Urf}(\text{Ulo})} \xi_2.pl_x \xrightarrow{\text{hb}} a \). From the construction of \( G.rf_1 \) we have \( G.rf_1 \subseteq G.po_1 \); moreover, from the construction of \( G.lo_1 \) we have \( G.lo_1 \subseteq G.po_1 \). We thus have \( \xi_2.pl_x \xrightarrow{po} b \). As such we have \( \xi_2.pl_x \xrightarrow{po} b \xrightarrow{\text{Urf}(\text{Ulo})} c \). Let \( c \in \theta_{\xi_a}.E \). There are now three cases to consider: a) \( b \xrightarrow{Urf} c \); or b) \( b \xrightarrow{rf} c \); or c) \( b \xrightarrow{lo} c \). We first demonstrate that in all three cases we have \( (\xi_1, \xi_a) \in \text{TCO} \).

In case (2.a), since \( \xi_2.pl_x \xrightarrow{po} c \), we know there exists \( c' \in T_{\xi_a} \) such that \( (w, c') \in G'.po_T \). As such, we have \( (r, c') \in G'.(rb_T; po_T) \). From the definition of TCO we thus have \( (\xi_1, \xi_a) \in \text{TCO} \). In case (2.b), from the definition of \( G.rf \) we know there exists \( c' \in T_{\xi_a} \) such that \( (w, c') \in G'.rf_T \). As such, we have \( (r, c) \in G'.(rb_T; rf_T) \). From the definition of TCO we thus have \( (\xi_1, \xi_a) \in \text{TCO} \). In case (2.c) from the construction of \( \text{lo} \) we know there exists \( z \) such that either \( b = \xi_2.pl_x \) or \( b = \xi_2.wu_z \), and that there exist \( p, q \) such that \( \xi_2 \in S^x_p, \xi_a \in S^x_q \) and \( p < q \). Given the definition of \( S^x \) we then know that there exists \( w_z \in G'.T_{\xi_2} \) where either i) there exists \( w'_z \in T_{\xi_a} \) such that \( (w_z, w'_z) \in G'.mo_T \); or ii) there exists \( r_z \in T_{\xi_a} \).
We proceed by contradiction. Let us assume that there exists $a, \theta \in G'$. That is, we have either $(w, w') \in G'.(\text{mot}; r_{T})$ or $(w, r_z) \in G'.(\text{mot}; r_{T})$. Moreover, since we have $(r, w) \in G'.(r_{b}; r_{T})$, we then have $(r, w') \in G', (r_{b}; r_{T})$, or $(r, r_z) \in G', (r_{b}; r_{T})$. From the definition of TCO we thus have $(\xi_1, \xi_c) \in \text{TCO}$.

Since $c \xrightarrow{b} a$, there are now two cases to consider: i) $c \xrightarrow{b} a$; or ii) $c \nrightarrow a$.

In case (2,i), from Lemma 11 we have $(c, a) \in p_{01} \lor (\xi_c, \xi_a) \in \text{TCO}$. As we have $(\xi_1, \xi_c) \in \text{TCO}$, we thus have $(\xi_1, \xi_a) \in \text{TCO}$, as required.

In case (2,ii) let us split the path from $c$ at the first occurrence of a non-\textit{hb} edge. That is, pick $\xi_3, \xi_4, g, h, y, p, q, s, k$ such that $c \rightarrow \xi_3, g \rightarrow \xi_4, h \rightarrow a$, $k < m$, $P_{\epsilon} \rightarrow \xi_3$, $P_{\epsilon} \rightarrow \xi_4$, $p > q$ and either a) $g = r_{\lambda} \land h = p_{\lambda}$; or b) $g = r_{\lambda} \land h = p_{\lambda}$; or $c) g = r_{\lambda} \land h = w_{\lambda}$; or d) $g = r_{\lambda} \land h = w_{\lambda}$.

From Lemma 11 we then have $(c, \xi_3, g) \in p_{01} \lor (\xi_3, \xi_4) \in \text{TCO}$. As we have $(\xi_1, \xi_c) \in \text{TCO}$, we also have $(\xi_1, \xi_3) \in \text{TCO}$. We next demonstrate that in all cases (2.ii.a-2.ii.d) there exists $t$ such that $t \leq m$ and $\xi_3, r_{u_{\lambda}} \rightarrow \xi_4, p_{l_{\lambda}} \xrightarrow{b} a$.

In case (2.ii.a) from the definition of $\text{lo}$ we also have $\xi_3, r_{u_{\lambda}} \rightarrow \xi_4, p_{l_{\lambda}} \xrightarrow{b} a$. In case (2.ii.b) the desired result holds immediately.

In cases (2.ii.c-2.ii.d) from the construction of $\text{lo}$ we have $\xi_3, r_{u_{\lambda}} \rightarrow \xi_4, p_{l_{\lambda}}$. Moreover, since we have $\xi_4, p_{l_{\lambda}} \rightarrow \xi_4, w_{u_{\lambda}}$, we also have $\xi_4, p_{l_{\lambda}} \rightarrow \xi_4, w_{u_{\lambda}} \xrightarrow{b} a$. As $p_{0} \subseteq \text{hb}$ and $\text{hb}$ is transitively closed, we have $\xi_4, p_{l_{\lambda}} \xrightarrow{\text{hb}^{-1}} a$. As such, we have $\xi_4, r_{u_{\lambda}} \rightarrow \xi_4, p_{l_{\lambda}} \xrightarrow{\text{hb}^{-1}} a$. As $k < m$, the desired result holds immediately.

Consequently, from (I.H.) we have $\text{stg}((\xi_3, r_{u_{\lambda}}) \rightarrow \text{stg}(a) \lor (\xi_3, \xi_a) \in \text{TCO}$. In the case of the first disjunct we have $\text{stg}((\xi_3, r_{u_{\lambda}}) = \text{stg}(\xi_3, r_{u_{\lambda}}) \leq \text{stg}(a)$, as required. In the case of the second disjunct, since we also have $(\xi_1, \xi_3) \in \text{TCO}$ and TCO is transitively closed, we have $(\xi_1, \xi_a) \in \text{TCO}$, as required.

**Theorem 10 (Completeness).** For all SI execution graphs $G'$ and their counterpart implementation graphs $G$ constructed as above,

$$\text{si-consistent}(G') \Rightarrow \text{RA-consistent}(G)$$

**Proof.** Pick an arbitrary SI execution graph $G'$ and its counterpart implementation graph $G$ constructed as above and assume \text{si-consistent}(G') holds. From the definition of \text{RA-consistent}(G) it suffices to show:

1. \text{irreflexive}(G.hb)
2. \text{irreflexive}(G.mot; G.hb)
3. \text{irreflexive}(G.rb; G.hb)

**RTS. part 1**

We proceed by contradiction. Let us assume that there exists $a, \theta \in G$. There are now two cases to consider: 1) $(a, a) \notin \text{hb}$; or 2) $(a, a) \notin \text{hb}$. 
In case (1), from Lemma 11 we have \((a, a) \in G.ppo \lor (T_\xi, T_\xi) \in TCO\). The first disjunct leads to a contradiction as the construction of \(G.ppo\) yields an acyclic relation. The second disjunct leads to a contradiction as \(TCO\) is a strict total order.

In case (2), let us split the \(a \rightarrow b\) at the first occurrence of a non-\(hb\)' edge. That is, pick \(\xi_1, \xi_2, i, j, k, g, h\) such that \(a \rightarrow \xi_1.g \rightarrow \xi_2.h \rightarrow b.a\). Let \(P^1_k \subseteq T_\xi\), \(P^2_j = T_\xi\) and \(j > k\). As we have \(\xi_1.g \rightarrow \xi_2.h \rightarrow a \rightarrow \xi_1.g\), from Lemma 12 and the definition of \(hb\) we then have \(stg(\xi_1.g) < \xi_1.g \lor (\xi_1, \xi_1) \in TCO\), leading to a contradiction in both disjuncts (the second disjunct yields a contradiction as \(TCO\) is a strict total order).

RTS. part 2
We proceed by contradiction. Let us assume that there exists \(a, \theta_\xi, b, \theta_\xi\) such that \(a \in \theta_\xi.E\), \(b \in \theta_\xi.E\), \((a, b) \in G.hb\) and \((b, a) \in G.mo\). Let \(loc(a) = loc(b) = x\) for some shared location \(x\). There are now two cases to consider: 1) \((b, a) \in G.mo\); or 2) \((b, a) \in G.mo\).

In case (1) we then have \((b, a) \in G'.mo \subseteq G'.po\). That is, we have \((b, a) \in G.ppo \subseteq G.hb\). We thus have \(a \rightarrow b \rightarrow a\), contradicting our proof in part 1. In case (2), from the construction of \(G.mo\) we have \((b, a) \in G'.mo\) and thus from the construction of \(G.lo\) we then have \((\xi_6.wux, \xi_6.rlx) \in G.lo\). As such we have \(a \rightarrow b \rightarrow \xi_6.wux \rightarrow \xi_6.rlx \rightarrow a\). That is, we have \(a \rightarrow a\), contradicting our proof in part 1.

RTS. part 3
We proceed by contradiction. Let us assume that there exists \(a, \theta_\xi, b, \theta_\xi\) such that \(a \in \theta_\xi.E\), \(b \in \theta_\xi.E\), \((a, b) \in G.hb\) and \((b, a) \in G.rb\).

Let \(loc(a) = loc(b) = x\) for some shared location \(x\). There are now two cases to consider: 1) \((b, a) \in G.rb\); or 2) \((b, a) \in G.rb\).

In case (1) we then have \((b, a) \in G'.rb \subseteq G'.po\). That is, we have \((b, a) \in G.ppo \subseteq G.hb\). We thus have \(a \rightarrow b \rightarrow a\), contradicting our proof in part 1. In case (2), from the construction of \(G.rb\) we have \((b, a) \in G'.rb\) and thus from the construction of \(G.lo\) we then have either \((\xi_6.wux, \xi_6.rlx) \in G.po\) and \((\xi_5.rux, \xi_5.rlx) \in G.lo\); or \((\xi_6.rux, \xi_5.rlx) \in po\). As such in both cases we have \((b, a) \in G.hb\). Consequently, we have \(a \rightarrow b \rightarrow a\), contradicting our proof in part 1.
E  Soundness and Completeness of the Eager RSI Implementation

Notation  Given an execution graph \((E, po, rf, mo, lo)\) we write \(T/st\) for the set of equivalence classes of \(T\) induced by \(st\); \([a]_{st}\) for the equivalence class that contains \(a\); and \(T_\xi\) for the equivalence class of transaction \(\xi \in TXIn: T_\xi \triangleq \{a | tx(a) = \xi\}\). We write \(rseq\triangleright\triangleright(G')\) to denote that \(G'\) is RSI-consistent; and write \(RA\triangleright\triangleright\triangleright(G)\) to denote that \(G\) is RA-consistent.

Given an execution graph \(G\) of the RSI implementation in Fig. 3, let us assign a transaction identifier to each transaction executed by the program; and given a transaction \(\xi\), let \(\text{RS}_\xi\) and \(\text{WS}_\xi\) denote its read and write sets, respectively. Observe that given a transaction \(\xi\) of the RSI implementation in Fig. 3 with \(\text{RS}_\xi \cup \text{WS}_\xi = \{x_1, \ldots, x_i\}\), the trace of \(\xi\), written \(\theta_\xi\), is of the following form:

\[
\theta_\xi = FS^* \xrightarrow{po|imm} Rs \xrightarrow{po|imm} RUs \xrightarrow{po|imm} PLs \xrightarrow{po|imm} Ts \xrightarrow{po|imm} Us
\]

where

- \(FS^*\) denotes the sequence of events failing to obtain a valid snapshot.
- \(Rs\) denotes the sequence of events acquiring a valid snapshot, and is of the form \(rl_{x_i} \xrightarrow{po|imm} \ldots \xrightarrow{po|imm} rl_{x_1} \xrightarrow{po|imm} S_{x_i} \xrightarrow{po|imm} \ldots \xrightarrow{po|imm} S_{x_1} \xrightarrow{po|imm} V_{x_i} \xrightarrow{po|imm} \ldots \xrightarrow{po|imm} V_{x_1}\), where for all \(n \in \{1 \cdots i\}\):

\[
rl_{x_n} = RL(x_n) = \begin{cases} RS_{x_n} \xrightarrow{po|imm} & \text{if } x \in \text{RS}_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]

\[
S_{x_n} = \begin{cases} ws_{x_n} \xrightarrow{po|imm} & \text{if } x \in \text{RS}_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]

\[
V_{x_n} = \begin{cases} vs_{x_n} & \text{if } x \in \text{RS}_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]

with \(rs_{x_n} \triangleq R(x_n, v_n)\), \(ws_{x_n} \triangleq W(s[x_n], v_n)\) and \(vs_{x_n} \triangleq R(x_n, v_n)\), for some \(v_n\).
- \(RUs\) denotes the sequence of events releasing the reader locks (when the given location is in the read set only), and is of the form \(ru_{x_i} \xrightarrow{po|imm} \ldots \xrightarrow{po|imm} ru_{x_1}\), where for all \(n \in \{1 \cdots i\}\):

\[
ru_{x_n} = \begin{cases} RU(x_n) & \text{if } x_n \not\in \text{WS}_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]

- \(PLs\) denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form \(pl_{x_i} \xrightarrow{po|imm} \ldots \xrightarrow{po|imm} pl_{x_1}\), where for all \(n \in \{1 \cdots i\}\):

\[
pl_{x_n} = \begin{cases} PL(x_n) & \text{if } x_n \in \text{WS}_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]
– $Ts$ denotes the sequence of events corresponding to the execution of $(T)$ in Fig. 3 and is of the form $t_1^{po|imm} \rightarrow \cdots \rightarrow t_k$, where for all $m \in \{1 \cdots k\}$:

$$t_m = \begin{cases} 
\{ R(s[x_n], v_n) \} & \text{if } O_m = R(x_n, v_n) \\
\{ w(x_n, v_n) \} & \text{if } O_m = W(x_n, v_n) 
\end{cases}$$

where $O_m$ denotes the $m$th event in the trace of the original $T$;

– $Us$ denotes the sequence of events releasing the locks on the write set. That is, the events in $Us$ correspond to the execution of the last line of the implementation in Fig. 3, and is of the form $wu_{x_1}^{po|imm} \rightarrow \cdots \rightarrow wu_{x_i}$, where for all $n \in \{1 \cdots i\}$:

$$wu_{x_n} = \begin{cases} 
\{ \overline{wu}(y_{1,n}) \} & \text{if } x_n \in \text{WS}_\xi \\
\emptyset & \text{otherwise} 
\end{cases}$$

Given a transaction trace $\theta_\xi$, we write e.g. $\xi.Ls$ to refer to its constituent $Ls$ sub-trace and write $Ls.E$ for the set of events related by $po$ in $Ls$. Similarly, we write $\xi.E$ for the set of events related by $po$ in $\theta_\xi$. Note that $G.E = \bigcup_{\xi \in Tx} \xi.E$.

### E.1 Implementation Soundness

In order to establish the soundness of our implementation, it suffices to show that given an RA-consistent execution graph of the implementation $G = (E, po, rf, mo, lo)$, we can construct a corresponding RSI-consistent execution graph $G'$ with the same outcome.

Given a transaction $\xi \in Tx$ with $RS_\xi \cup WS_\xi = \{x_1 \cdots x_i\}$ and trace $\theta_\xi = F_{s^*}^{po|imm} R_{s^{po|imm}} R_{Us}^{po|imm} PL_{s}^{po|imm} T_{s}^{po|imm} Us$, with $Ts = t_1^{po|imm} \rightarrow \cdots \rightarrow t_k$, we define $\theta'_\xi \triangleq t'_1^{po|imm} \rightarrow \cdots \rightarrow t'_k$, such that for all $m \in \{1 \cdots k\}$:

$$t'_m = R(x_n, r_{bn}) \text{ when } t_m = R(s[x_n], r_{bn})$$

$$t'_m = W(x_n, r_{bn}) \text{ when } t_m = W(s[x_n], r_{bn})$$

such that in the first case the identifier of $t'_m$ is that of $t_m$; and in the second case the identifier of $t'_m$ is that of the first event in $t_m$. We then define:

$$RF_\xi \triangleq \left\{ \begin{array}{l}
(t'_j, t'_j) \in Ts_\xi^* \land \exists x, v. t'_j = R(x, v) \land w = W(x, v) \\
\land (w \in \xi.E \Rightarrow w \xRightarrow{po} t'_j) \\
\land (\forall e \in \xi.E. w \xRightarrow{po} e \xRightarrow{po} t'_j \Rightarrow (loc(e) \neq x \lor e \notin WS)) \\
\land (w \notin \xi.E \Rightarrow (\forall e \in \xi.E. e \xRightarrow{po} t'_j \Rightarrow (loc(e) \neq x \lor e \notin WS)) \\
\land (w, \xi, rs_k), (w, \xi, vs_k) \in G.rf)
\end{array} \right\}$$

We are now in a position to demonstrate the soundness of our implementation. Let $G.NT \triangleq G.E \setminus \bigcup_{\xi \in Tx} \theta_\xi.E$. Given an RA-consistent execution graph $G$ of the implementation, we construct an RSI execution graph $G'$ as follows and demonstrate that $\text{rsi-consistent}(G')$ holds.
On the other hand, from Lemma 2 we know that either i) case (iii) the desired result holds trivially.

Similarly in case (ii) we have \(a, b, \xi\) such that \(\xi_a \neq \xi_b\), \(a \in \xi_a.E\), \(b \in \xi_b.E\), and \(\text{loc}(a) = \text{loc}(b) = x\).

Lemma 13. Given an RA-consistent execution graph \(G\) of the implementation and its corresponding RSI execution graph \(G'\) constructed as above, for all \(a, b, \xi_a, \xi_b, x\):

\[
\xi_a \neq \xi_b \land a \in \xi_a.E \land b \in \xi_b.E \land \text{loc}(a) = \text{loc}(b) = x \implies
\]

\[
((a, b) \in G'.rf \implies \xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x) \quad (22)
\]

\[
\land ((a, b) \in G'.mo \implies \xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x) \quad (23)
\]

\[
\land ((a, b) \in G'.mo; rf \implies \xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x) \quad (24)
\]

\[
\land ((a, b) \in G'.rb \implies (x \in \text{WS}_{\xi_a} \land \xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x) \lor (x \notin \text{WS}_{\xi_a} \land \xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.pl_x)) \quad (25)
\]

Proof. Pick an arbitrary RA-consistent execution graph \(G\) of the implementation and its corresponding RSI execution graph \(G'\) constructed as above. Pick an arbitrary \(a, b, \xi_a, \xi_b, x\) such that \(\xi_a \neq \xi_b\), \(a \in \xi_a.E\), \(b \in \xi_b.E\), and \(\text{loc}(a) = \text{loc}(b) = x\).

RTS. (22)
Assume \((a, b) \in G'.rf\). From the definition of \(G'.rf\) we then know \((a, \xi_a.rs_x), (a, \xi_a.ws_x) \in G.rf\). On the other hand, from Lemma 2 we know that either i) \(x \in \text{WS}_{\xi_a}\) and \(\xi_b.ws_x \xrightarrow{G_{bb}} \xi_a.rl_x\); or ii) \(x \notin \text{WS}_{\xi_b}\) and \(\xi_b.rs_x \xrightarrow{G_{bb}} \xi_a.pl_x\); or iii) \(\xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x\). In case (i) we then have \(a \xrightarrow{G^{rf}} \xi_a.rs_x \xrightarrow{G^{po}} \xi_b.ws_x \xrightarrow{G^{bb}} \xi_a.pl_x \xrightarrow{G^{po}} a\). That is, we have a contradiction, contradicting the assumption that \(G\) is RA-consistent.

Similarly in case (ii) we have \(a \xrightarrow{G^{rf}} \xi_a.rs_x \xrightarrow{G^{po}} \xi_b.rs_x \xrightarrow{G^{bb}} \xi_a.pl_x \xrightarrow{G^{po}} a\). That is, we have a contradiction, contradicting the assumption that \(G\) is RA-consistent. In case (iii) the desired result holds trivially.

RTS. (23)
Assume \((a, b) \in G'.mo\). From the definition of \(G'.mo\) we then know \((a, b) \in G.mo\).

On the other hand, from Lemma 2 we know that either i) \(\xi_b.ws_x \xrightarrow{G_{bb}} \xi_a.rl_x\); or ii) \(\xi_a.ws_x \xrightarrow{G_{ab}} \xi_b.rl_x\). In case (i) we then have \(a \xrightarrow{G^{mo}} b \xrightarrow{G^{po}} \xi_b.ws_x \xrightarrow{G^{bb}} \xi_a.pl_x \xrightarrow{G^{po}} a\).
That is, we have \( a \xrightarrow{G,\text{hb}_0} a \), contradicting the assumption that \( G \) is RA-consistent. In case (ii) the desired result holds trivially.

**RTS. (24)**

Assume \( (a, b) \in G'.\text{mo;rf} \). We then know there exists \( w \) such that \( (a, w) \in G'.\text{mo} \) and \( (w, b) \in G'.\text{rf} \). From the definition of \( G'.\text{mo} \) we then know \( (a, w) \in G'.\text{mo} \). There are now two cases to consider: 1) \( w \in \xi_b \); or 2) \( w \notin \xi_b \). In case (1) since \( (a, w) \in G'.\text{mo} \) the desired result follows from part 23.

In case (2) from the definition of \( G'.\text{rf} \) we know that \( (w, \xi_b, rs_x) \in G'.\text{rf} \).

From Lemma 2 we know that either i) \( x \in WS_{\xi_b} \) and \( \xi_b, wu_x \xrightarrow{G,\text{hb}} \xi_a, rl_x \); or ii) \( x \notin WS_{\xi_b} \) and \( \xi_b, ru_x \xrightarrow{G,\text{hb}} \xi_a, pl_x \); or iii) \( \xi_a, wu_x \xrightarrow{G,\text{hb}} \xi_a, rl_x \). In case (i) we then have \( a \xrightarrow{G,\text{mo}} w \xrightarrow{G,\text{rf}} \xi_a, rs_x \xrightarrow{G,\text{po}} \xi_a, rl_x \xrightarrow{G,\text{hb}} a \). That is, we have \( a \xrightarrow{G,\text{mo}} w \xrightarrow{G,\text{rf}} a \), contradicting the assumption that \( G \) is RA-consistent. Similarly in case (ii) we have \( a \xrightarrow{G,\text{mo}} w \xrightarrow{G,\text{rf}} \xi_a, rs_x \xrightarrow{G,\text{po}} \xi_b, ru_x \xrightarrow{G,\text{hb}} \xi_a, pl_x \xrightarrow{G,\text{po}} a \). That is, we have \( a \xrightarrow{G,\text{mo}} w \xrightarrow{G,\text{hb}} a \), contradicting the assumption that \( G \) is RA-consistent.

In case (iii) the desired result holds trivially.

**RTS. (25)**

Assume \( (a, b) \in G'.\text{rb} \). From the definition of \( G'.\text{rb} \) we know \( (\xi_a, rs_x, b), (\xi_a, vs_x, b) \in G'.\text{rb} \). On the other hand, from Lemma 2 we know that either i) \( \xi_b, wu_x \xrightarrow{G,\text{hb}} \xi_a, rl_x \); or ii) \( x \notin WS_{\xi_b} \) and \( \xi_a, ru_x \xrightarrow{G,\text{hb}} \xi_a, pl_x \); or iii) \( x \in WS_{\xi_a} \) and \( \xi_a, wu_x \xrightarrow{G,\text{hb}} \xi_b, rl_x \). In case (i) we then have \( b \xrightarrow{G,\text{po}} \xi_b, wu_x \xrightarrow{G,\text{hb}} \xi_a, rl_x \xrightarrow{G,\text{po}} \xi_a, rs_x \xrightarrow{G,\text{rb}} b \). That is, we have \( b \xrightarrow{G,\text{po}} \xi_a, rs_x \xrightarrow{G,\text{rb}} b \), contradicting the assumption that \( G \) is RA-consistent.

In cases (ii-iii) the desired result holds trivially.

Let

\[ \text{rsi-hb'} \triangleq \text{rsi-hb} \setminus \{ \text{si-rb}; \text{rsi-hb} \} \]

**Lemma 14.** For all RA-consistent execution graphs \( G \) of the implementation and their counterpart RSI execution graphs \( G' \) constructed as above,

1. \( (G',\text{rsi-po} \subseteq G.\text{hb}) \land (G',\text{(rsi-po;si-rb)} \subseteq G.\text{hb}) \)
2. \( (G',\text{rsi-mo} \subseteq G.\text{hb}) \land (G',\text{(rsi-mo;si-rb)} \subseteq G.\text{hb}) \)
3. \( (G',\text{rsi-rf} \subseteq G.\text{hb}) \land (G',\text{(rsi-rf;si-rb)} \subseteq G.\text{hb}) \)
4. \( (G',\text{rsi-hb'} \subseteq G.\text{hb}) \)

**Proof (Proof (Part 1)).** Pick an arbitrary RA-consistent execution graph \( G \) of the implementation and its counterpart RSI execution graph \( G' \) constructed as above. Pick arbitrary \( (a, b) \in G'.\text{rsi-po} \) and \( (b, c) \in G'.\text{si-rb} \). There are then two cases to consider: 1) \( (a, b) \notin G'.\text{rsi-po} \); or 2) \( (a, b) \in G'.\text{rsi-po} \).
In case (1) from the construction of $\text{rsi-po}$ we then have $(a,b) \in G_{\text{po}} \subseteq G_{\text{hb}}$, as required by the first conjunct. For the second conjunct, we know there exists $\xi, r, w$ such that $b, r \in \xi, R_E, c, w \in \xi, W$, and $(r, w) \in G_{\text{rb}}$. Let $\text{loc}(r) = \text{loc}(w) = x$. Given the definition of $R_E$ and the construction of $G'$ we know $(\xi_b, rs_x, w) \in G_{\text{rb}}$.

As we have $(a,b) \notin G'_{\text{rsi-po}}$ and $b$ and $\xi_b, rs_x$ are both transactional events in $\xi_b$, from the definition of $G'_{\text{po}}$ and $G_{\text{po}}$ we then know $(a, \xi_b, rs_x) \in G_{\text{po}}$. On the other hand, from Lemma 13 we have either i) $x \in WS_{\xi_b}$ and $\xi_b, ru_x \rightarrow \xi_b, pl_x$, or ii) $x \notin WS_{\xi_b}$ and $\xi_b, ru_x \rightarrow \xi_c, pl_x$. In case (i) we have $a \rightarrow rs_x \rightarrow c$, i.e. $a \rightarrow b$, as required. Similarly, in case (ii) have $a \rightarrow rs_x \rightarrow c$, i.e. $a \rightarrow c$, as required.

In case (2) from the definition of $G'_{\text{rsi-po}}$ we then know that $a, b \in G'/W$ and thus from the construction of $G_{\text{rsi-po}}$ we have $(a,b) \in G_{\text{po}} \subseteq G_{\text{hb}}$, as required by the first conjunct. For the second conjunct, since we have $b \in G'/W$, we cannot have $(b,c) \in G'_{\text{si-rb}}$ and thus the desired result holds vacuously.

Proof (Proof (Part 2)). Pick an arbitrary RA-consistent execution graph $G$ of the implementation and its counterpart RSI execution graph $G'$ constructed as above. For the first conjunct pick arbitrary $(a,b) \in G'_{\text{rsi-mo}}$. From the definition of $G'_{\text{rsi-mo}}$ we then know there exists $w, w', \xi, \xi_b$ such that $\xi \neq \xi_b, w, a \in \xi, E, w', b \in \xi_b, E$ and $(w, w') \in G_{\text{mo}}$. From the definition of $G'_{\text{mo}}$ we then have $(w, w') \in G_{\text{po}}$. Let $\text{loc}(w) = \text{loc}(w') = x$. From Lemma 13 we then know that $\xi_{a,w} \rightarrow G_{\text{po}} \rightarrow G_{\text{hb}} \rightarrow b$, i.e. $a \rightarrow b$, as required.

For the second conjunct pick arbitrary $c$ such that $(b,c) \in G_{\text{si-rb}}$. From the definition of $\text{si-rb}$ and the construction of $G'$ we then know there exist $y, r, y', \xi, \xi_c$ such that $\xi \neq \xi_c, \text{loc}(w) = \text{loc}(r) = y, (r, y) \in G_{\text{rb}}$, and $w, c \in \xi_c, W$. As we demonstrated for the first conjunct we have $a \rightarrow rs_x \rightarrow c$, i.e. $a \rightarrow c$, as required. Similarly, in case (ii) we have $a \rightarrow rs_x \rightarrow c$, i.e. $a \rightarrow c$, as required.

Proof (Proof (Part 3)). Pick an arbitrary RA-consistent execution graph $G$ of the implementation and its counterpart RSI execution graph $G'$ constructed as above. It suffices to show that:

\begin{align*}
G'([\mathcal{N}T]; \mathcal{R}f; \mathcal{S}t) & \subseteq G_{\text{hb}} \quad (26) \\
G'([\mathcal{N}T]; \mathcal{R}f; \mathcal{S}t); G'_{\text{si-rb}} & \subseteq G_{\text{hb}} \quad (27)
\end{align*}
\[ G'.rf_{\top} \subseteq G.hb \]  
\[ G'.rf_{\top}; G'.si-rb \subseteq G.hb \]  
\[ G'.(mo; rf)_{\top} \subseteq G.hb \]  
\[ G'.(mo; rf)_{\top}; G'.si-rb \subseteq G.hb \]  
\[ G'.(rf; [N{T}]) \subseteq G.hb \]  
\[ G'.(rf; [N{T}]); G'.si-rb \subseteq G.hb \]  

\textbf{RTS. (26)}

Pick arbitrary \((w, a) \in G'.([N{T}]; rf; st)\) where \(a \in \xi_a.E\). Let \(\text{loc}(w) = x\). From the construction of \(G'.rf\) we then know \((w, \xi_a.rs_x), (w, \xi_a.vs_x) \in G.rf\) and \((\xi_a.rs_x, a), (\xi_a.vs_x, a) \in G.po\). As such we have \(w \xrightarrow{G'.rf} \xi_a.rs_x \xrightarrow{G'.po} a\), i.e. \(w \xrightarrow{G.hb} a\), as required.

\textbf{RTS. (27)}

Pick arbitrary \((w, c) \in G'.([N{T}]; rf; st); G'.si-rb\). We then know there exist \(a, \xi_a, \xi_c\) such that \((w, a) \in G'.([N{T}]; rf; st)\), \((a, c) \in G'.si-rb\). \(a \in \xi_a.E, c \in \xi_c.W\) and \(\xi_a \neq \xi_c\). Let \(\text{loc}(w) = y\). As we demonstrated in the previous part we then know \((w, \xi_a.rs_{x_y}), (w, \xi_a.vs_{x_y}) \in G.rf\) and \((\xi_a.rs_x, a), (\xi_a.vs_x, a) \in G.po\). Moreover, from the definition of \(si-rb\) and the construction of \(G'\) we know there exist \(y, y, w_y, \xi_c\) such that \(\text{loc}(w_y) = \text{loc}(r_{y_y}) = y\), \((r_y, w_y) \in G'.rb\), and \(w_y, c \in \xi_c.W\). From the construction of \(G'.rb\) we then know that \((\xi_a.rs_{y_y}, w_y), (\xi_a.vs_{y_y}, w_y) \in G.rb\). On the other hand, from Lemma 13 we know that either \(i) y \notin WS_{\xi_c}\) and \(\xi_{a.ws_{y_y}} \xrightarrow{G.hb} \xi_{c.rl_{y_y}}\); or \(ii) y \in WS_{\xi_c}\) and \(\xi_{a.ws_y} \xrightarrow{G.hb} \xi_{c.pl_y}\). In case \(i\) we have \(w \xrightarrow{G'.rf} \xi_{a.rs_x} \xrightarrow{G'.po} \xi_{a.vs_y} \xrightarrow{G'.po} \xi_{a.ws_{y_y}} \xrightarrow{G.hb} \xi_{b.rl_{y_y}} \xrightarrow{G'.po} c\), i.e. \(a \xrightarrow{G.hb} c\), as required. Similarly, in case \(ii\) we have \(w \xrightarrow{G'.rf} \xi_a.rs_x \xrightarrow{G'.po} \xi_a.vs_y \xrightarrow{G'.po} \xi_{a.ws_{y_y}} \xrightarrow{G.hb} \xi_{b.pl_y} \xrightarrow{G'.po} c\), i.e. \(w \xrightarrow{G.hb} c\), as required.

\textbf{RTS. (28)}

Pick arbitrary \((a, b) \in G'.rf_{\top}\). We then know there exist \(w, r, \xi_a, \xi_b\) such that \(w, a \in \xi_a.E, r, b \in \xi_b.E, \xi_a \neq \xi_b\) and \((w, r) \in G'.rf\). Let \(\text{loc}(w) = \text{loc}(r) = x\). From Lemma 13 we then know that \(\xi_a.ws_{w_x} \xrightarrow{G.hb} \xi_{b.rl_x}\). Consequently from the structure of \(G\) and the construction of \(G'\) we have \(a \xrightarrow{G'.po} \xi_{a.ws_{w_x}} \xrightarrow{G.hb} \xi_{b.rl_x} \xrightarrow{G'.po} b\), i.e. \(a \xrightarrow{G.hb} b\), as required.

\textbf{RTS. (29)}

Pick arbitrary \((a, c) \in G'.rf_{\top}; G'.si-rb\). \(b\) such that \((a, b) \in G'.rf_{\top}\) and \((b, c) \in G'.si-rb\). From the definition of \(G'.rf_{\top}\) we then know there exist \(w, r, \xi_a, \xi_b\) such that \(w, a \in \xi_a.E, r, b \in \xi_b.E, \xi_a \neq \xi_b\) and \((w, r) \in G'.rf\). Let \(\text{loc}(w) = \text{loc}(r) = x\). From Lemma 13 we then know that \(\xi_a.ws_{w_x} \xrightarrow{G.hb} \xi_{b.rl_x}\).

On the other hand, from the definition of \(G'.si-rb\) we know there exist \(r', w', \xi_c\) such that \(r' \in \xi_b.E, w', c \in \xi_c.W, \xi_c \neq \xi_b\) and \((r', w') \in G'.rb\). Let \(\text{loc}(w') = x\).
\( \text{loc}(r') = y. \) From Lemma 13 we then know that either i) \( y \in \text{WS}_{\xi_E} \) and \( \xi, w, u_y \rightarrow_{G, b} \xi_c, r_l y \) or ii) \( y \not\in \text{WS}_{\xi_E} \) and \( \xi, r_uy \rightarrow_{G, b} \xi_c, p_l y \).

In case (i) we have a \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, w_y} \xi_c, r_l y \rightarrow_{G, po} c. \) That is, \( (a, c) \in G, h, b \), as required. Similarly, in case (ii) we have a \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, r_u y} \xi_c, p_l y \rightarrow_{G, po} c. \) This is, \( (a, c) \in G, h, b \), as required.

RTS. (30)
Pick arbitrary \( (a, b) \in G'.(\text{mo}; r_f)_T \). We then know there exist \( \xi, \xi, w, w', r \) such that \( w, a \in \xi_E, r, b \in \xi_E, \xi \neq \xi, (w, w') \in G', \text{mo} \) and \( (w', r) \in G', r_f \).

From Lemma 13 we then have \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, w_y} \xi_c, r_l y \rightarrow_{G, po} b, \) i.e. \( a \rightarrow_{G, h} b \), as required.

RTS. (31)
Pick arbitrary \( (a, c) \in G'.(\text{mo}; r_f)_T; G'. \text{si-rb} \). We then know there exist \( b \) such that \( (a, b) \in G'.(\text{mo}; r_f)_T \) and \( (b, c) \in G'. \text{si-rb} \). From the definition of \( G'.(\text{mo}; r_f)_T \) we then know there exist \( \xi, \xi, w, w', r \) such that \( w, a \in \xi_E, r, b \in \xi_E, \xi \neq \xi, (w, w') \in G', \text{mo} \) and \( (w', r) \in G', r_f \). From Lemma 13 we then have \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, r_u y} \xi_c, r_l y \rightarrow_{G, po} c. \)

On the other hand, from the definition of \( G'. \text{si-rb} \) we know there exist \( r'' \), \( w''; \xi \) such that \( r'' \in \xi_E, w'' \in \xi_E, \xi \neq \xi, (r'', w'') \in G', \text{rb} \). Let \( \text{loc}(w'') = \text{loc}(r'') = y. \) From Lemma 13 we then know that either i) \( y \not\in \text{WS}_{\xi_E} \) and \( \xi, w, u_y \rightarrow_{G, h} \xi_c, r_l y \) or ii) \( y \not\in \text{WS}_{\xi_E} \) and \( \xi, r_u y \rightarrow_{G, h} \xi_c, p_l y \).

In case (i) we then have a \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, w_y} \xi_c, r_l y \rightarrow_{G, po} c. \) That is, \( (a, c) \in G, h, b \), as required. Similarly, in case (ii) we then have a \( \xi_a, w_x \rightarrow_{G, h} \xi_b, r_l x \rightarrow_{\xi, r_u y} \xi_c, p_l y \rightarrow_{G, po} c. \) That is, \( (a, c) \in G, h, b \), as required.

RTS. (32)
Pick arbitrary \( (w, r) \in G'.(r_f; [N_T]). \) As \( r \in G', [N_T] \), from the construction of \( G', r_f \) we have \( (w, r) \in G, r_f \subseteq G, h, b \), as required.

RTS. (33)
The desired result holds trivially as \( G'.(r_f; [N_T]); G', \text{si-rb} = \emptyset. \)

Proof (Proof part 4). Let \( rsi-hb_0 = rsi-po \cup \text{rsi-rf} \cup \text{rsi-mo} \cup \text{si-rb} \) and \( rsi-hb_{n+1} = rsi-hb_0, rsi-hb_n \), for all \( n \geq 0 \). Similarly, let \( rsi-hb'_0 = rsi-hb_0 \setminus \text{si-rb} \) and \( rsi-hb'_{n+1} = rsi-hb'_0, rsi-hb'_n \), for all \( n \geq 0 \). It is then straightforward to demonstrate that \( rsi-hb' = \bigcup_{n \in N} rsi-hb'_n \). It thus suffices to show that:

\[ \forall n \in \mathbb{N}, rsi-hb'_n \subseteq G, h \]
We proceed by induction on \( n \).

**Base case** \( n = 0 \)

Follows immediately from the definition of \( rsi-hb'_0 \) and the results established in 1-3.

**Inductive case** \( n = m+1 \)

\[ \forall i \in \mathbb{N}, i < n \Rightarrow rsi-hb'_i \subseteq G.hb \]  
(I.H.)

Pick arbitrary \((a, b) \in rsi-hb'_n\). From the definition of \( rsi-hb'_n \), we then know there exists \( c \) such that \((a, c) \in rsi-hb'_0\) and \((c, b) \in rsi-hb'_m\). Let \( rsi-hb'_{-1} \triangleq \text{id} \). There are now two cases to consider: 1) \((c, b) \in rsi-hb'_0; rsi-hb_{m-1} \); or \((c, b) \in si-rb; rsi-hb_{m-1} \).

In case (1) from the proof of base case we have \((a, c) \in G.hb\). On the other hand from the definition of \( rsi-hb'_m \) we have \((c, b) \in rsi-hb'_m\) and thus from (I.H.) we have \((a, b) \in G.hb\). Consequently, since \( G.hb \) is transitively closed we have \((a, b) \in G.hb\) as required.

In case (2) we know there exists \( d \) such that \((c, d) \in si-rb\); and \((d, b) \in rsi-hb_{m-1}\). Since we have \((a, c) \in rsi-hb'_0\) and \((c, d) \in si-rb\), from the definition of \( rsi-hb'_0 \) and the proofs of parts 1-3 we have \((a, d) \in G.hb\). Moreover we either have i) \( m = 0 \); or ii) \( m > 0 \). In case (2.i) since \((d, b) \in rsi-hb_{m-1}\), from the definition of \( rsi-hb_{m-1} \) we have \( b = d \) and thus \((a, b) \in G.hb\), as required. In case (2.ii) from (I.H.) we then have \((d, b) \in G.hb\). As such, since \( G.hb \) is transitively closed, we have \((a, b) \in G.hb\) as required.

**Lemma 15.** For all RA-consistent execution graphs \( G \) of the implementation and their counterpart RSI execution graphs \( G' \) constructed as above:

\[ \forall n \in \mathbb{N}. \forall a, b, w, \xi, x. \]
\[ (a, b) \in G'.rsi-hb_n \land a \in G'.W \land b \in G'.(R \cap T_\xi) \Rightarrow \]
\[ (w \in G'.(W \cap T_\xi) \Rightarrow (a, w) \in G.hb) \land (\xi vs_x \text{ defined } \Rightarrow (a, \xi vs_x) \in G.hb) \]

where \( rsi-hb_0 \triangleq rsi-po \cup rsi-rf \cup rsi-mo \cup si-rb \) and \( rsi-hb_{n+1} \triangleq rsi-hb'_n; rsi-hb_0 \), for all \( n \geq 0 \).

**Proof.** Pick an arbitrary RA-consistent execution graph \( G \) of the implementation and its counterpart RSI execution graph \( G' \) constructed as above. We then proceed by induction on \( n \).

**Base case** \( n = 0 \)

Pick arbitrary \( a, b, \xi, x \) such that \((a, b) \in (rsi-hb_0), a \in G'.W, b \in G'.(R \cap T_\xi), \xi vs_x \text{ defined } \) and \( w \in G'.(W \cap T_\xi) \). From the definition of \( rsi-hb_0 \) we then have either 1) \((a, b) \in G'.rsi-po \land a \not\in G'.T_\xi \); or 2) \((a, b) \in G'.(mo_T \cup rf_T \cup (mo; rf)_T) \); or 3) \((a, b) \in G'.(\{N_T\}; rf); \) and thus \((a, \xi vs_x), (a, w) \in G.po \subseteq G.hb\), as required.

In case (1) from the definition of \( G'.rsi-po \) we then know that \( \{a\} \times \theta_\xi.E \subseteq G.po \) and thus \((a, \xi vs_x), (a, w) \in G.po \subseteq G.hb\), as required.

In case (2) from the definitions of \( rf_T, mo_T, (mo; rf)_T \) we know there exists \( \xi', c, d, y \) such that \( a, c \in G'.T_{\xi'}, a, c \in \xi'.E, b, d \in G'.T_\xi, b, d \in \xi.E, (c, d) \in \).
We proceed by contradiction. Let us assume there exists a case to consider: 1) \( c \) does not include any internal edges to read events. In case (1) from Lemma 14 part 4 we then have as required. Similarly we have as required.

In case (3) from the construction of \( G' \) we know there exists \( y \) such that \( \text{loc}(a) = y \), and \((a, \xi.rs_y), (a, \xi.vs_x) \in G, rf \). As such we have \( a \xrightarrow{G,bb} \xi.rs_y \xrightarrow{G,po} \xi.vs_x \), i.e. \( a \xrightarrow{G,bb} w \), as required. Moreover, since we have \( \xi.rs_y \xrightarrow{G,po} w \), we have \( a \xrightarrow{G,bb} w \), as required.

**Inductive case** \( n = m+1 \)

Pick arbitrary \( a, b, \xi, x \) such that \((a, b) \in (\text{rsi-hb}_n), a \in G', W, b \in G', (R \cap T_\xi) \), \( \xi.vs_x \) defined and \( w \in G', (W \cap T_\xi) \). From the definition of \( \text{rsi-hb}_n \) we then know there exists \( c \) such that \((a, c) \in \text{rsi-hb}_m \) and \((b, c) \in \text{rsi-hb}_0 \). There are then two cases to consider: 1) \( c \in T_\xi \); or 2) \( c \notin T_\xi \).

Case (1) leads to contradiction as \((c, b) \in \text{rsi-hb}_0 \), \( b \in T_\xi \), \( b \notin R \) and \( \text{rsi-hb}_0 \) does not include any internal edges to read events.

In case (2) since \( a \) is a write event, from the definition of \( \text{rsi-hb}' \) we then have \((a, c) \in \text{rsi-hb}' \) and thus from Lemma 14 we have \((a, c) \in G, \text{hb} \). On the other hand, since \( c \notin T_\xi \), \((b, c) \in \text{rsi-hb}_0 \) and \( b \) is a read event, from the definition of \( \text{rsi-hb}_0 \) we know that either 1) \((b, c) \in G', \text{rsi-po} \land c \notin T_\xi \); or 2) \((b, c) \in G', (\text{mo} \cup r_{tf} \cup (\text{mo};rf) \cup \text{rt} \cup \text{st}) \); or 3) \((b, c) \in G', ([N_T]; \text{rsi-po}) \). Following an analogous argument as that in the base case, we then have \( c \xrightarrow{G,bb} \xi.vs_x \) and \( c \xrightarrow{G,bb} w \). As such, we have \( a \xrightarrow{G,bb} c \xrightarrow{G,bb} \xi.vs_x \), i.e. \( a \xrightarrow{G,bb} \xi.vs_x \), as required. Similarly, we have \( a \xrightarrow{G,bb} w \), i.e. \( a \xrightarrow{G,bb} w \), as required.

**Theorem 11 (Soundness).** For all execution graphs \( G \) of the implementation and their counterpart RSI execution graphs \( G' \) constructed as above,

\[
\text{RA-consistent}(G) \Rightarrow \text{rsi-consistent}(G')
\]

**Proof.** Pick an arbitrary execution graph \( G \) of the implementation such that \( \text{RA-consistent}(G) \), and its associated RSI execution graph \( G' \) constructed as described above. It then suffices to show 1) \( \text{rsi-hb}_0 \cup \text{mo} \cup r_{tf} \subseteq \text{po} \); 2) \( \text{acyclic}(G', \text{rsi-hb}_{loc}); \) 3) \( \text{acyclic}(G', (\text{rsi-hb}_{loc}; \text{mo}); \) and 4) \( \text{acyclic}(G', (\text{rsi-hb}_{loc}; \text{rb}); \)

**RTS.** \( \text{rf} \cup \text{mo} \cup \text{rb} \subseteq \text{po} \)

Follows immediately from the construction of \( G' \) and the RA-consistency of \( G \).

**RTS.** \( \text{acyclic}(G', \text{rsi-hb}_{loc}) \)

We proceed by contradiction. Let us assume there exists such that \((a, a) \in G', \text{rsi-hb}_{loc} \). There are then two cases to consider: i) \((a, a) \in \text{rsi-hb}' \); or ii) \((a, a) \in \text{rsi-hb} \setminus \text{rsi-hb}' \). In case (1) from Lemma 14 part 4 we then have \((a, a) \in G, \text{hb} \), contradicting the assumption that \( G \) is RA-consistent. In case (2) we
We proceed by contradiction. Let us assume there exist \(a, b \in G, W\), \((a, b) \in si-rb\) and \((b, a) \in rsi-hb\). Moreover, since \(b \in G, W\), from the definition of \(si-rb\) and \(rsi-hb\) we have \((b, a) \in rsi-hb'\). Since we also have \((a, b) \in si-rb \subseteq rsi-hb\), from the definition \(rsi-hb'\) we have \((b, a) \in rsi-hb'\). Consequently, from Lemma 14 part 4 we have \((b, b) \in G, hb\), contradicting the assumption that 
\(G\) is RA-consistent.

\[RTS. \quad \text{acyclic}(G', (rsi-hb_{loc}; mo))\]

We proceed by contradiction. Let us assume there exist \(a, b \in G', rsi-hb_{loc}\) and \((a, b) \in G', mo\). From the definition of \(G', mo\) we then know that \(a, b \in G', W\) and \((a, b) \in G, mo\). On the other hand, since \(a \in G', W\) and \((a, b) \in rsi-hb\), from the definition of \(rsi-hb'\) we have \((a, b) \in rsi-hb'\). Consequently, from Lemma 14 part 4 we have \((a, b) \in G, hb\). We then have \(a \xrightarrow{G, hb} b \xrightarrow{G, mo} a\), contradicting the assumption that 
\(G\) is RA-consistent.

\[RTS. \quad \text{acyclic}(G', (rsi-hb_{loc}; rb))\]

We proceed by contradiction. Let us assume there exist \(w, r \in G', rsi-hb_{loc}\) and \((w, r) \in G', rb\). From the definition of \(G', rb\) we then know that \(w \in G', W\) and \(r \in G', R\). Since \(w \in G', W\), from the definition of \(rsi-hb'\) we have \((w, r) \in rsi-hb'\). Consequently, from Lemma 14 part 4 we have \((w, r) \in G, hb\). Let \(loc(w) = loc(r) = x\). There are then two cases to consider: 1) \(r \in G', NT\); or 2) \(\exists \xi_r, r \in \xi_r, E\).

In case (1) from the definition of \(G', rb\) we know \((r, w) \in G, rb\). As such, we have \(w \xrightarrow{G, rb} r \xrightarrow{G, rb} w\), contradicting the assumption that \(G\) is RA-consistent.

In case (2) we then know there exists \(w'\) such that \((w', r) \in G', rf\) \(w', w) \in G', mo\). There are now three cases to consider: a) \(w \in \xi_r, E\); or b) \(w \not\in \xi_r, E \land w' \in \xi_r, E\); or c) \(w, w' \not\in \xi_r, E\). In case (2.a) from the construction of \(G'\) (in particular, \(G', rf\)) we know that \((r, w) \in G', po_1 \subseteq G, po_1 \subseteq G, hb\). As such, we have \(w \xrightarrow{G, hb} r \xrightarrow{G, hb} w\), contradicting our assumption that 
\(G\) is RA-consistent.

In case (2.b), since \(w'\) is a write event, \(w' \in \xi_r, E\), \(r\) is a read event, \(w\) is a write event and \((w, r) \in G', rsi-hb\), from Lemma 15 we have \((w, w') \in G, hb\). Moreover, since \((w, w) \in G', mo\), from the definition of \(G', mo\) we also have \((w', w) \in G, mo\). As such we have \(w \xrightarrow{G, rb} w' \xrightarrow{G, mo} w\), contradicting the assumption that \(G\) is RA-consistent.

In case (2.c), from the construction of \(G'\) we then know \((\xi_r, rs_{w^2}, w) \in G, rb\) and \((\xi_r, vs_{w^2}, w) \in G, rb\). On the other hand, since \(r\) is a read event, \(w\) is a write event and \((w, r) \in G', rsi-hb\), from Lemma 15 we have \((w, \xi_r, vs_{w^2}) \in G, hb\). As such we have \(w \xrightarrow{G, rb} \xi_r, vs_{w^2} \xrightarrow{G, rb} w\), contradicting the assumption that 
\(G\) is RA-consistent.

E.2 Implementation Completeness

In order to establish the completeness of our implementation, it suffices to show that given an RSI-consistent execution graph \(G' = (E, po, rf, mo, lo)\), we can construct a corresponding RA-consistent execution graph \(G\) of the implementation. Before proceeding with the construction of a corresponding implementation graph, we describe several auxiliary definitions.
Given a transaction class \( \mathcal{T}_\xi \in G'.\mathcal{T}/st \), we write \( \mathcal{WS}_\xi \) for the set of locations written to by \( \mathcal{T}_\xi \). We similarly write \( \mathcal{RS}_\xi \) for the set of locations read from by \( \mathcal{T}_\xi \). For each location \( x \) read from by \( \mathcal{T}_\xi \), we additionally record the first read event in \( \mathcal{T}_\xi \) that retrieved the value of \( x \). That is,

\[
\mathcal{RS}_\mathcal{T}_\xi \triangleq \left\{ (x, r) \middle| r \in \mathcal{T}_\xi \land x \in \mathcal{R}_x \land \exists e \in \mathcal{T}_\xi \land e \xrightarrow{\text{po}} r \right\}
\]

Note that the execution trace for each transaction \( \mathcal{T}_\xi \in G'.\mathcal{T}/st \) is of the form \( \theta'_\xi = t'_1 \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} t'_k \), comprising a series of read or write events. As such, we have \( G'.E = G'.\mathcal{T} = \bigcup_{\mathcal{T}_\xi \in G'.\mathcal{T}/st} \mathcal{T}_\xi = \theta'_\xi \cdot E \). Let \( \mathcal{RS}_\mathcal{T}_\xi \cup \mathcal{WS}_\mathcal{T}_\xi = \{x_1 \cdots x_n\} \). We then construct the implementation trace \( \theta_\xi \) as:

\[
\theta_\xi = R_s \xrightarrow{\text{po|imm}} R_u_s \xrightarrow{\text{po|imm}} P_l_s \xrightarrow{\text{po|imm}} t_s \xrightarrow{\text{po|imm}} U_s
\]

where

- \( R_s = r_{x_1} \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} r_{x_n} \xrightarrow{\text{po|imm}} S_{x_1} \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} S_{x_n} \xrightarrow{\text{po|imm}} V_{x_1} \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} V_{x_n} \), where the identifiers of all constituent events of \( R_s \) are picked fresh, and

\[
rl_{x_j} = \begin{cases}
\text{RL}(x_j) & \text{if } (x_j, r) \in \mathcal{RS}_\xi \land \text{val}(r) = v_j \\
\emptyset & \text{otherwise}
\end{cases}
\]

\[
S_{x_j} = \begin{cases}
\text{us}_{x_j} & \text{if } (x_j, r) \in \mathcal{RS}_\xi \land \text{val}(r) = v_j \\
\emptyset & \text{otherwise}
\end{cases}
\]

\[
V_{x_j} = \begin{cases}
\text{us}_{x_j} & \text{if } (x_j, r) \in \mathcal{RS}_\xi \land \text{val}(r) = v_j \\
\emptyset & \text{otherwise}
\end{cases}
\]

with \( r_{x_j} \triangleq R(x_j, v_j) \), \( \text{us}_{x_j} \triangleq R(x_j, v_j) \) and \( \text{us}_{x_j} \triangleq W(s[x_j], v_j) \).

- \( R_u_s = r_{u_{x_1}} \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} r_{u_{x_n}} \), where the identifiers of all constituent events of \( R_u_s \) are picked fresh, and for all \( j \in \{1 \cdots n\} \):

\[
r_{u_{x_j}} = \begin{cases}
\text{RU}(x_j) & \text{if } x_j \notin \mathcal{WS}_\xi \\
\emptyset & \text{otherwise}
\end{cases}
\]

- \( P_l_s = p_{l_{x_1}} \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} p_{l_{x_n}} \), where the identifiers of all constituent events of \( P_l_s \) are picked fresh, and for all \( j \in \{1 \cdots n\} \):

\[
p_{l_{x_j}} = \begin{cases}
\text{PL}(x_j) & \text{if } x_j \in \mathcal{WS}_\xi \\
\emptyset & \text{otherwise}
\end{cases}
\]

- \( t_s = t_1 \xrightarrow{\text{po|imm}} \cdots \xrightarrow{\text{po|imm}} t_k \), where for all \( m \in \{1 \cdots k\} \):

\[
t_m = \begin{cases}
\text{R}(s[x_n], v_n) & \text{if } t_m' = \text{R}(x_n, v_n) \\
\text{W}(x_n, v_n) & \text{if } t_m' = \text{W}(x_n, v_n)
\end{cases}
\]
such that in the first case the identifier of \( t_m \) is that of \( t'_m \); and in the second case the identifier of the first event in \( t_m \) is that of \( t'_m \) and the identifier of the second event is picked fresh.

\[-Us = \overset{\text{po}}{w_{x_1}} \rightarrow \cdots \overset{\text{po}}{w_{x_n}}, \text{ where the identifiers of all constituent events of } Us \text{ are picked fresh, and} \]

\[
w_{x_j} = \begin{cases} \mathbf{wU}(x_j) & \text{if } x_j \in WS_\xi \\ \emptyset & \text{otherwise} \end{cases}
\]

In what follows we write \( E_\xi \) as a shorthand for the events in the implementation trace of \( \Theta_\xi \), i.e. \( E_\xi \triangleq \{ a \mid a \in \Theta_\xi.E \} \). We use the \( \xi \) prefix to project the various events of the implementation trace \( \Theta_\xi \) (e.g. \( \xi.rl_\xi \)).

For each location \( x \) we then define:

\[
\text{LO}_x \triangleq \{ (\xi, rl_x, \xi, pl_x), (\xi, rl_x, \xi, wu_x), (\xi, pl_x, \xi, wu_x) \mid G'.T_\xi \cap WS_x \neq \emptyset \} \\
\bigcup \{ (\xi, rl_x, \xi', pl_x), (\xi, rl_x, \xi', wu_x), (\xi, pl_x, \xi', pl_x), (\xi, pl_x, \xi', wu_x), (\xi, ru_x, \xi', ru_x), (\xi, ru_x, \xi', wu_x) \mid \xi \neq \xi' \land \exists a, b, x. \}
\]

\[
\bigcup \{ (\xi, rl_x, \xi, pl_x), (\xi, rl_x, \xi', wu_x), (\xi, ru_x, \xi, pl_x), (\xi, ru_x, \xi', pl_x), (\xi, ru_x, \xi, wu_x), (\xi, ru_x, \xi', wu_x) \mid \xi \neq \xi' \land \exists a, b, x. \}
\]

\[
\bigcup \{ (\xi, pl_x, \xi', rl_x), (\xi, pl_x, \xi', ru_x), (\xi, pl_x, \xi', wu_x), (\xi, pl_x, \xi', ru_x) \mid \xi \neq \xi' \land \exists a, b, x. \}
\]

\[
\bigcup \{ (w, \xi, rl_x), (w, \xi, ru_x), (w, \xi, wu_x) \mid \exists r. (x, r) \in RS_\xi \land (w, r) \in G'rf \} \cup \{ G'rf \cap G'.E \times G'.N'T \}
\]

\[
\text{LO}_x = \bigcup_{x \in \text{LOCS}} \text{LO}_x, \text{ with } \text{LO}_x \text{ as defined above.}
\]
Notation  In what follows, given an RSI implementation graph $G$ as constructed above we write $G.NT$ for the non-transactional events of $G$, i.e. $G.NT \triangleq \{ a \mid \\exists \xi. a \in G.E_\xi \}$. Moreover, as before, given a relation $r \subseteq G.E \times G.E$, we override the $r_1$ notation and write $r_1$ for $\{ (a, b) \mid \exists \xi. a, b \in \theta_\xi.E \}$.

Lemma 16. Given an RSI-consistent execution graph $G'$ and its corresponding implementation graph $G$ constructed as above, for all $a, b, \xi, \xi_0$:

$$(a, b) \in G.hb \Rightarrow \exists \xi, a, b \in G.E_\xi \Rightarrow (a, b) \in G.po$$

$$\wedge (\exists \xi, a, b \in G.E_\xi \Rightarrow$$

$$\exists A, B, \emptyset \subset A \times B \subseteq G'.hsi-hb$$

$$\wedge (a \in G.NT \Rightarrow A=\{a\}) \wedge (b \in G.NT \Rightarrow B=\{b\})$$

$$\wedge [a \in G.E_\xi \Rightarrow$$

$$A=G'.T_\xi \vee (\text{stg}(a, \xi_a) \leq 2 \wedge A=G'.T_\xi \cap R_e)$$

$$\vee (\text{stg}(a, \xi_a) \leq 4 \wedge \exists d \in \xi_a.W_a \stackrel{G.po}{\rightarrow} d \wedge A=\{d\})$$

$$\wedge [b \in G.E_\xi \Rightarrow$$

$$B=G'.T_\xi \vee (\text{stg}(b, \xi_b) \geq 3 \wedge B=G'.T_\xi \cap W)]$$

where

$$\text{stg}(a, \xi_a) \triangleq \begin{cases} 
1 & \text{if } a \in \xi_a.Rs_\xi \\
2 & \text{if } a \in \xi_a.RUs_\xi \\
3 & \text{if } a \in \xi_a.PLs_\xi \\
4 & \text{otherwise}
\end{cases}$$

Proof. Pick an arbitrary RSI-consistent execution graph $G'$ and its corresponding implementation graph $G$ constructed as above. Let $hb_0 \triangleq G.(po \cup rf \cup lo)$ and $hb_{n+1} \triangleq hb_n \cup hb_n$, for all $n \in \mathbb{N}$. It is then straightforward to demonstrate that $G.hb = \bigcup_{n \in \mathbb{N}} hb_n$. We thus demonstrate instead that:

$$\forall n \in \mathbb{N}. \forall a, b, \xi_a, \xi_b. (a, b) \in G.hb_n \Rightarrow$$

$$\exists \xi, a, b \in G.E_\xi \Rightarrow (a, b) \in G.po$$

$$\wedge (\exists \xi, a, b \in G.E_\xi \Rightarrow$$

$$\exists A, B, \emptyset \subset A \times B \subseteq G'.hsi-hb$$

$$\wedge (a \in G.NT \Rightarrow A=\{a\}) \wedge (b \in G.NT \Rightarrow B=\{b\})$$

$$\wedge [a \in G.E_\xi \Rightarrow$$

$$A=G'.T_\xi \vee (\text{stg}(a, \xi_a) \leq 2 \wedge A=G'.T_\xi \cap R_e)$$

$$\vee (\text{stg}(a, \xi_a) \leq 4 \wedge \exists d \in \xi_a.W_a \stackrel{G.po}{\rightarrow} d \wedge A=\{d\})$$

$$\wedge [b \in G.E_\xi \Rightarrow$$

$$B=G'.T_\xi \vee (\text{stg}(b, \xi_b) \geq 3 \wedge B=G'.T_\xi \cap W)]$$

We proceed by induction on $n$. 

There are three cases to consider: 1) \((a, b) \in G.\text{po}\); or 2) \((a, b) \in G.\text{rf}\); or 3) \((a, b) \in G.\text{lo}\).

In case (1) there are five cases to consider: a) \(\exists \xi. (a, b) \in G.E_\xi\); or b) \(a, b \in G.\text{NT}\); or c) \(a \in G.\text{NT}\) and \(b \in G.E_\xi\); or d) \(a \in G.E_\xi\) and \(b \in G.\text{NT}\); or e) \(a \in G.E_\xi\), \(b \in G.E_\xi\), and \(\xi_a \neq \xi_b\). In case (1.a) we then have \((a, b) \in G.\text{po}\), as required. In case (1.b) from the construction of \(G.\text{po}\) we have \((a, b) \in G'.\text{po}\), as required. In case (1.c) from the construction of \(G.\text{po}\) we have \((\{a\} \times G'.\mathcal{T}_\xi) \subseteq G'.\text{po}\), as required. In case (1.d) from the construction of \(G.\text{po}\) we have \((G'.\mathcal{T}_\xi \times \{b\}) \subseteq G'.\text{po}\), as required. In case (1.e) from the construction of \(G.\text{po}\) we have \((G'.\mathcal{T}_\xi \times G'.\mathcal{T}_\xi) \subseteq G'.\text{po}\), as required.

In case (2) there are five cases to consider: a) \(\exists \xi. (a, b) \in G.E_\xi\); or b) \(a, b \in G.\text{NT}\); or c) \(a \in G.\text{NT}\) and \(b \in G.E_\xi\); or d) \(a \in G.E_\xi\) and \(b \in G.\text{NT}\); or e) \(a \in G.E_\xi\), \(b \in G.E_\xi\), and \(\xi_a \neq \xi_b\). Case (2.a) holds vacuously as \((a, b) \in G.\text{rf}_1 = \emptyset\). In case (2.b) from the construction of \(G.\text{rf}\) we have \((a, b) \in G'.\text{rf}\), as required. In case (2.c) from the construction of \(G.\text{rf}\) we know that there exists \(r \in G'.\mathcal{T}_\xi\) such that \((a, r) \in G'.\text{rf}\). As such we have \((\{a\} \times G'.\mathcal{T}_\xi) \subseteq G'.([\text{NT}]; \text{rf}; \text{st}) \subseteq G'.\text{rsi-hb}\), as required. In case (2.d) from the construction of \(G.\text{rf}\) we then have \((a, b) \in G'.\text{rf}\); that is a \(G.\text{po}\) and \((a, b) \in G'.\text{rf}\), as required. In case (2.e) from the construction of \(G.\text{rf}\) we know that there exists \(r \in G'.\mathcal{T}_\xi\) such that \((a, r) \in G'.\text{rf}\). As such we have \((G'.\mathcal{T}_\xi \times G'.\mathcal{T}_\xi) \subseteq G'.\text{rf}\), as required.

In case (3) there are five cases to consider: a) \(\exists \xi. (a, b) \in G.E_\xi\); or b) \(a, b \in G.\text{NT}\); or c) \(a \in G.\text{NT}\) and \(b \in G.E_\xi\); or d) \(a \in G.E_\xi\) and \(b \in G.\text{NT}\); or e) \(a \in G.E_\xi\), \(b \in G.E_\xi\), and \(\xi_a \neq \xi_b\). In case (3.a) from the construction of \(G.\text{lo}\) we have \((a, b) \in G.\text{po}\), as required. Cases (3.b-3.d) hold vacuously as there are no \text{lo} edge to or from non-transactional events. In case (3.e) from the construction of \(G.\text{lo}\) we know there exist \(x, c, d\) such that \(c \in G'.\mathcal{T}_\xi\), \(d \in G'.\mathcal{T}_\xi\), and either i) \((c, d) \in G'.\text{rf}\); or ii) \((c, d) \in G'.(\text{mo}; \text{rf})\); or iii) \((c, d) \in G'.\text{mo}\); or iv) \(\text{loc}(c) = \text{loc}(d) = x, a = \xi_a.r_t \lor \xi_a.t_x \lor \xi_a.r_u_x, b = \xi_a.w_t \lor \xi_a.w_u_x, c = G'.\mathcal{R}_E\) and \((c, d) \in G'.\text{rb}\). In cases (3.e.i, 3.e.ii) we have \(G'.\mathcal{T}_\xi \times G'.\mathcal{T}_\xi \subseteq G'.([\text{rf} \cup (\text{mo}; \text{rf})]; \text{st}) \subseteq \text{rsi-rf} \subseteq \text{rsi-hb}\), as required. In case (3.e.iii) we have \(G'.\mathcal{T}_\xi \times G'.\mathcal{T}_\xi \subseteq G'.\text{mo} \subseteq \text{rsi-hb}\), as required. In case (3.e.iv) we then have \(\text{stg}(a, \xi_a) \leq 2, \text{stg}(b, \xi_b) \geq 3,\) and \(G'.(\mathcal{T}_\xi \cap \mathcal{R}_E) \times G'.(\mathcal{T}_\xi \cap \mathcal{W}) \subseteq G'.\text{si-rb} \subseteq \text{rsi-hb}\), as required.
Inductive case \( n = m+1 \)

\[ \forall i \in \mathbb{N}, \forall a, b, \xi_a, \xi_b, i \leq m \land (a, b) \in G.hb_i \Rightarrow \\
(\exists \xi. a, b \in G.E_{\xi} \Rightarrow (a, b) \in G.po) \\
\land \left( \neg \exists \xi. a, b \in G.E_{\xi} \Rightarrow \\
\exists A, B. \emptyset \subset A \times B \subseteq G', \text{rsi-hb} \\
\land (a \in G.N'T \Rightarrow A = \{a\}) \land (b \in G.N'T \Rightarrow B = \{b\}) \\
\land [a \in G.E_{\xi_a} \Rightarrow \\
A = G'.T_{\xi_a} \lor (stg(a, \xi_a) \leq 2 \land A = G'.T_{\xi_a} \cap R_E) \\
\lor (stg(a, \xi_a) \leq 4 \land \exists d \in \xi_a, W. a \xrightarrow{G,N'T} d \land A = \{d\})] \\
\land [b \in G.E_{\xi_b} \Rightarrow \\
B = G'.T_{\xi_b} \lor (stg(b, \xi_b) \geq 3 \land B = G'.T_{\xi_b} \cap W)] \right) \] (I.H.)

Since \((a, b) \in hb_n\), from the definition of \(hb_n\) we know there exists \(c\) such that \((a, c) \in hb_0\) and \((c, b) \in hb_m\). There are then five cases to consider: 1) \(\exists \xi. (a, b) \in G.E_{\xi}\); or 2) \(a, b \in G.N'T\); or 3) \(a \in G.N'T\) and \(b \in G.E_{\xi_a}\); or 4) \(a \in G.E_{\xi_a}\) and \(b \in G.N'T\); or 5) \(a \in G.E_{\xi_a}, b \in G.E_{\xi_b}\) and \(\xi_a \neq \xi_b\).

Case 1

In case (1) pick arbitrary \(\xi\) such that \(a, b \in G.E_{\xi}\). There are then three additional cases to consider: a) \(c \in G.N'T\); b) \(c \in G.E_{\xi}\); or c) there exists \(\xi' \neq \xi\) such that \(c \in G.E_{\xi'}\).

In case (1.a) from the proof of the base case and the (I.H.) we know there exist \(A, B, C \neq \emptyset\) such that \(C = \{c\}, A \times C \subseteq G'.\text{rsi-hb}, C \times B \subseteq G'.\text{rsi-hb}\) and thus \(A \times B \subseteq G'.\text{rsi-hb}\) and either: i) \(A = G'.T_{\xi} and B = G'.T_{\xi}\); or ii) \(A = G'.T_{\xi}\) and \(B = (G'.(T_{\xi} \cap W))\); or iii) \(stg(a, \xi) \leq 2, A = G'.(T_{\xi} \cap R_E)\) and \(B = G'.T_{\xi}\); or iv) \(stg(a, \xi) \leq 4, \exists d \in \xi, W. a \xrightarrow{G,N'T} d \land A = \{d\}\) and \(B = G'.T_{\xi}\); or v) \(stg(a, \xi) \leq 4, \exists d \in \xi, W. a \xrightarrow{G,N'T} d \land A = \{d\}\) and \(B = G'.(T_{\xi} \cap W)\).

Case (i) cannot arise as we would have \(A = B\) and thus \(A \times A \subseteq G'.\text{rsi-hb}\), contradicting the assumption that \(G'\) is RSI-consistent. Case (ii) cannot arise as we would have \((G'.(T_{\xi} \cap W)) \times (G'.(T_{\xi} \cap W)) \subseteq G'.\text{rsi-hb}\), contradicting the assumption that \(G'\) is RSI-consistent. Case (iii) cannot arise as we would have \((G'.(T_{\xi} \cap R_E) \times (G'.(T_{\xi} \cap R_E)) \subseteq G'.\text{rsi-hb}\), contradicting the assumption that \(G'\) is RSI-consistent. In case (iv) since we have \(stg(a, \xi) \leq 2\) and \(stg(b, \xi) \geq 3\), from the definition of \(stg(\_)\) and the construction of \(G\) we have \((a, b) \in G.po\), as required. Cases (v-vi) cannot arise as we would have \(\exists d \in \xi, W. (d, d) \in G'.\text{rsi-hb}\), contradicting the assumption that \(G'\) is RSI-consistent.

In case (1.b) from the proof of the base case we have \((a, c) \in G.po\). On the other hand from (I.H.) we have \((c, b) \in G.po\). As \(G.po\) is transitively closed, we have \((a, b) \in G.po\), as required.
In case (1.c) from the proof of the base case (in cases 1.e, 2.e and 3.e) we have either A) $G',\mathcal{T}_\xi \times G',\mathcal{T}_\xi \subseteq G'.\text{rsi-hb}$; or B) $\text{stg}(a,\xi) \leq 2$, $\text{stg}(c,\xi) \geq 3$, and $\emptyset \subset (\mathcal{T}_\xi \cap \mathcal{R}_E) \times G'.(\mathcal{T}_\xi \cap \mathcal{W}) \subset G'.\text{rsi-hb}$. On the other hand from (I.H.) we know there exist $C, B \neq \emptyset$ such that $C \times B \subsetneq G'.\text{rsi-hb}$ and either: i) $C = G',\mathcal{T}_\xi$ and $B = G',\mathcal{T}_\xi$; or ii) $C = G',\mathcal{T}_\xi$, $\text{stg}(b,\xi) \geq 3$ and $B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$; or iii) $\text{stg}(c,\xi) \leq 2$, $C = G'.(\mathcal{T}_\xi \cap \mathcal{R}_E)$, $\text{stg}(b,\xi) \geq 3$ and $B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$; or iv) $\text{stg}(c,\xi) \leq 4$, $\exists d \in \xi'.\mathcal{W}$. c $\xrightarrow{G,po}$ $d \cap C = \{d\}$ and $B = G'.\mathcal{T}_\xi$; or vi) $\text{stg}(c,\xi') \leq 4$, $\exists d \in \xi'.\mathcal{W}$. c $\xrightarrow{G,po}$ $d \cap C = \{d\}$, $\text{stg}(b,\xi) \geq 3$ and $B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$.

Cases (A.i-A.vi) lead to a cycle in $G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent. In cases (B.ii, B.iv, B.vi) we then have $\text{stg}(a,\xi) \leq 2$ and $\text{stg}(b,\xi) \geq 3$. Consequently from the definition of $\text{stg}(\cdot,\cdot)$ and the construction of $G$ we have $(a,b) \in G.\text{po}$, as required. Case (B.i) leads to a cycle in $G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent. In cases (B.ii) we have $\text{stg}(c,\xi') \geq 3$ and $\text{stg}(c,\xi') \leq 2$, leading to a contradiction. In case (B.vi) we then know $\exists d \in \xi'.\mathcal{W}, B \subset G'.(\mathcal{T}_\xi \cap \mathcal{R}_E) \times \{d\} \subset G'.\text{rsi-hb} \cap \{d\} \times G'.\mathcal{T}_\xi \subseteq G'.\text{rsi-hb}$. That is, we have $\emptyset \subset G'.(\mathcal{T}_\xi \cap \mathcal{R}_E) \times G'.(\mathcal{T}_\xi \cap \mathcal{R}_E) \subseteq G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent.

Case 2
There are two additional cases to consider: a) $c \in G.\mathcal{N}T$; or b) there exists $\xi$ such that $c \in G.E_\xi$.

In case (2.a) from the proof of the base case we have $(a,c) \in G'.\text{rsi-hb}$. On the other hand from (I.H.) we have $(c,b) \in G'.\text{rsi-hb}$. As $G'.\text{rsi-hb}$ is transitively closed, we have $(a,b) \in G'.\text{rsi-hb}$, as required.

In case (2.b) from the proof of the base case we know there exist $C_1 \neq \emptyset$ such that $\{a\} \times C_1 \subseteq G'.\text{rsi-hb}$ and either: A) $C_1 = G'.\mathcal{T}_\xi$; or B) $\text{stg}(c,\xi) \geq 3$ and $C_1 = G'.(\mathcal{T}_\xi \cap \mathcal{W})$. On the other hand, from (I.H.) we know there exist $C_2 \neq \emptyset$ such that $C_2 \times \{b\} \in G'.\text{rsi-hb}$ and either: i) $C_2 = G'.\mathcal{T}_\xi$; or ii) $\text{stg}(c,\xi) \leq 2$ and $C_2 = G'.(\mathcal{T}_\xi \cap \mathcal{R}_E)$; or iii) $\text{stg}(c,\xi) \leq 4$ and $\exists d \in \xi'.\mathcal{W}. c \xrightarrow{G,po} d \cap C_2 = \{d\}$.

In cases (A.i-A.iii) and (B.i, B.iii) from the transitivity of $G'.\text{rsi-hb}$ we have $(a,b) \in G'.\text{rsi-hb}$, as required. Case (B.ii) cannot arise as otherwise we would have $3 \leq \text{stg}(c,\xi) \leq 2$, leading to a contradiction.

Case 3
There are three additional cases to consider: a) $c \in G.\mathcal{N}T$; b) $c \in G.E_\xi$; or c) there exists $\xi_\delta \neq \xi_\delta$ such that $c \in G.E_\xi$.

In case (3.a) from the proof of the base case we have $(a,c) \in G'.\text{rsi-hb}$. On the other hand from (I.H.) we know there exist $B \neq \emptyset$ such that $\{c\} \times B \subseteq G'.\text{rsi-hb}$ and $B = G'.\mathcal{T}_\xi \vee (\text{stg}(b,\xi_b) \geq 3 \wedge B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$. As $G'.\text{rsi-hb}$ is transitively closed we then know there exist $B \neq \emptyset$ such that $\{a\} \times B \subseteq G'.\text{rsi-hb}$ and $B = G'.\mathcal{T}_\xi \vee (\text{stg}(b,\xi_b) \geq 3 \wedge B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$, as required.

In case (3.b) from the proof of the base case we know there exist $B \neq \emptyset$ such that $\{a\} \times B \subseteq G'.\text{rsi-hb}$ and $B = G'.\mathcal{T}_\xi \vee (\text{stg}(c,\xi_c) \geq 3 \wedge B = G'.(\mathcal{T}_\xi \cap \mathcal{W})$. On
the other hand, from (I.H.) we have \( c \xrightarrow{G,po} b \) and thus from the definition of \( \text{stg}(\ldots) \) and the construction of \( G \) we have \( \text{stg}(b, \xi_b) \geq \text{stg}(c, \xi_c) \). As such, we know there exists \( B \neq \emptyset \) such that \( \{a\} \times B \subseteq G'.rsi-hb \) and \( B = G'.T_{\xi_b} \vee (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W)) \), as required.

In case (3.c) from the proof of the base case we know there exists \( C_1 \neq \emptyset \) such that \( \{a\} \times C_1 \subseteq G'.rsi-hb \) and either A) \( C_1 = G'.T_{\xi_b} \); or B) \( \text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_b} \cap W) \). On the other hand, from (I.H.) we know there exist \( C_2, B \neq \emptyset \) such that \( C_2 \times B \subseteq G'.rsi-hb \) and either: i) \( C_2 = G'.T_{\xi_b} \) and \( B = G'.T_{\xi_b} \); or ii) \( C_2 = G'.T_{\xi_b} \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W) \); or iii) \( \text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_b} \cap R_E) \) and \( B = G'.T_{\xi_b} \); or iv) \( \text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_b} \cap R_E) \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W) \); or v) \( \text{stg}(c, \xi_c) \leq 4 \land \exists d \in \xi_c.W. c \xrightarrow{G,po} d \land C_2 = \{d\} \) and \( B = G'.T_{\xi_b} \); or vi) \( \text{stg}(c, \xi_c) \leq 4 \land \exists d \in \xi_c.W. c \xrightarrow{G,po} d \land C_2 = \{d\} \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W) \).

In cases (A.i-A.vi) (B.i, B.ii, B.v, B.vi) from the transitivity of \( G'.rsi-hb \) we know there exists \( B \neq \emptyset \) such that \( \{a\} \times B \subseteq G'.rsi-hb \) and \( B = G'.T_{\xi_b} \vee (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W)) \), as required. Cases (B.iii, B.iv) cannot arise as we would otherwise have \( 3 \leq \text{stg}(c, \xi_c) \leq 2 \), leading to a contradiction.

**Case 4**

There are three additional cases to consider: a) \( c \in G.N'T \); b) \( c \in G.E_{\xi_a} \); or c) there exists \( \xi_a \neq \xi_c \) such that \( c \in G.E_{\xi_a} \).

In case (4.a) from (I.H.) we have \( \{c, b\} \subseteq G'.rsi-hb \). On the other hand from the base case we know there exists \( A \neq \emptyset \) such that \( A \times \{c\} \subseteq G'.rsi-hb \) and \( A = G'.T_{\xi_a} \vee (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \vee \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \). As \( G'.rsi-hb \) is transitively closed we then know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.rsi-hb \) and \( A = G'.T_{\xi_a} \vee (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \vee \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \), as required.

In case (4.b) from (I.H.) we know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.rsi-hb \) and \( A = G'.T_{\xi_a} \vee (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \vee \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \). On the other hand from the proof of the base case we have \( a \xrightarrow{G,po} c \) and thus from the definition of \( \text{stg}(\ldots) \) and the construction of \( G \) we have \( \text{stg}(a, \xi_a) \leq \text{stg}(c, \xi_c) \). As such, we know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.rsi-hb \) and \( A = G'.T_{\xi_a} \vee (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \vee \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \), as required.

In case (4.c) from the proof of the base case we know there exist \( A, C_1 \neq \emptyset \) such that \( A \times C_1 \subseteq G'.rsi-hb \) and either: i) \( A = G'.T_{\xi_a} \) and \( C_1 = G'.T_{\xi_b} \); or ii) \( A = G'.T_{\xi_a} \) and \( \text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_b} \cap W) \); or iii) \( \text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \) and \( C_1 = G'.T_{\xi_b} \); or iv) \( \text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \) and \( \text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_b} \cap W) \); or v) \( \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \) and \( C_1 = G'.T_{\xi_b} \); or vi) \( \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,po} d \land A = \{d\} \) and \( \text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_b} \cap W) \).
On the other hand, from \((I.H.)\) we know there exists \(C_2 \neq \emptyset\) such that \(C_2 \times \{b\} \subseteq G'.\text{rsi-hb}\) and either \(A\) \(C_2 = G'.T_{\xi_a}\); or \(B\) \((\text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_c} \cap R_E))\); or \(C\) \((\text{stg}(c, \xi_c) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} c \land C_2 = \{e\})\).

In cases \((A.i-A.vi), (C.i-C.vi)\) and \((B.i, B.iii, B.v)\) from the transitivity of \(G'.\text{rsi-hb}\) we know there exist \(A \neq \emptyset\) such that \(A \times \{b\} \subseteq G'.\text{rsi-hb}\) and \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\), as required. Cases \((B.ii, B.iv, B.vi)\) cannot arise as we would otherwise have \(3 \leq \text{stg}(c, \xi_c) \leq 2\), leading to a contradiction.

**Case 5**

There are four additional cases to consider: \(a\) \(c \in G.N.T\); \(b\) \(c \in G.E_{\xi_a}\); or \(c\) \(c \in G.E_{\xi_b}\); or \(d\) there exists \(\xi\) such that \(\xi_c \neq \xi_a\), \(\xi_c \neq \xi_b\) and \(c \in G.E_{\xi_c}\).

In case \((5.a)\) from the proof of the base case we know there exists \(A \neq \emptyset\) such that \(A \times \{c\} \subseteq G'.\text{rsi-hb}\) and \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\). On the other hand from \((I.H.)\) we know there exists \(B \neq \emptyset\) such that \(\{c\} \times B \subseteq G'.\text{rsi-hb}\) and \(B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))\). As such, since \(G'.\text{rsi-hb}\) is transitive we know there exist \(A, B \neq \emptyset\) such that \(A \times B \subseteq G'.\text{rsi-hb}\); that \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\); and that \(B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))\), as required.

In case \((5.b)\) from \((I.H.)\) we know there exist \(A, B \neq \emptyset\) such that \(A \times B \subseteq G'.\text{rsi-hb}\); that \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(c, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\); and that \(B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))\), as required.

In case \((5.c)\) from the proof of the base case we know there exist \(A, B \neq \emptyset\) such that \(A \times B \subseteq G'.\text{rsi-hb}\); that \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\); and that \(B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))\), as required.

In case \((5.d)\) from the proof of the base case we know there exist \(A, C_1 \neq \emptyset\) such that \(A \times C_1 \subseteq G'.\text{rsi-hb}\); that \(A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists \xi \in \xi_a.W. a \xrightarrow{G'.\text{po}} d \land A = \{d\})\); and that \(C_1 = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))\), as required.
We proceed by contradiction. Let us assume that there exists an (a,ξ) \in R_E \cup \{a\} \cup \{d\}, and that either A) C_1 = G'.T_{\xi}; or B) \text{stg}(c,\xi) \geq 3 \wedge C_1 = G'.(T_{\xi} \cap W).

On the other hand, from (I.H.) we know that there exist C_2, B \neq \emptyset such that C_2 \times B \subseteq G'.rsgi-hb; that B = G'.T_{\xi} \cup \text{stg}(b,\xi) \geq 3 \wedge B = G'.(T_{\xi} \cap W); and that either: i) C_2 = G'.T_{\xi}; or ii) \text{stg}(c,\xi) \leq 2 \wedge C_2 = G'.(T_{\xi} \cap R_E); iii) \text{stg}(c,\xi) \leq 4 \wedge 3d \in \xi; W; c \stackrel{G'.po}{\rightarrow} d \wedge C_2 = \{d\}).

In cases (A.i-A.iii) and (B.i, B.iii) from the transitivity of G'.rsgi-hb we know there exists A, B \neq \emptyset such that A \times B \subseteq G'.rsgi-hb; that A = G'.T_{\xi} \cup \text{stg}(a,\xi) \leq 2 \wedge A = G'.(T_{\xi} \cap R_E) \cup \text{stg}(a,\xi) \leq 4 \wedge 3d \in \xi; W; a \stackrel{G'.po}{\rightarrow} d \wedge A = \{d\}); and that \text{stg}(b,\xi) \geq 3 \wedge B = G'.(T_{\xi} \cap W)), as required. Case (B.ii) cannot arise as we would otherwise have 3 \leq \text{stg}(c,\xi) \leq 2, leading to a contradiction.

**Theorem 12 (Completeness).** For all RSI execution graphs G’ and their counterpart implementation graphs G constructed as above,

\[
\text{rsi-consistent}(G') \Rightarrow \text{RA-consistent}(G)
\]

Proof. Pick an arbitrary RSI execution graph G’ and its counterpart implementation graph G constructed as above and let us assume that rsi-consistent(G’) holds. From the definition of RA-consistent(G) it then suffices to show:

1. irreflexive(G hb)
2. irreflexive(G.mo; G hb)
3. irreflexive(G.rb; G hb)

**RTS. part 1**

We proceed by contradiction. Let us assume that there exists a such that (a, a) \in G hb. There are now two cases to consider: 1) a \in G NT; or 2) \exists \xi. a \in G E_{\xi}

In case (1) from Lemma 16 we have (a, a) \in G’.rsgi-hb, contradicting the assumption that G’ is RSI-consistent. Similarly, in case (2) from Lemma 16 we have (a, a) \in G po, leading to a contradiction as G po is acyclic by construction.

**RTS. part 2**

We proceed by contradiction. Let us assume that there exist a, b such that (a, b) \in G hb and (b, a) \in G mo. From the construction of G mo we then know that (b, a) \in G’.mo. Let \text{loc}(a) = \text{loc}(b) = x. There are then five cases to consider: 1) \exists \xi. (a, b) \in G E_{\xi}; or 2) a, b \in G NT; or 3) a \in G NT and b \in G E_{\xi}; or 4) a \in G E_{\xi} and b \in G NT; or 5) a \in G E_{\xi}, b \in G E_{\xi}, and \xi \neq \xi.

In case (1) from Lemma 16 we have (a, b) \in G po, and since a, b \in G W = G’ W, from the construction of G po we have (a, b) \in G’ po. Moreover, since a, b are write events in the same transaction \xi, (a, b) \in G’ po1 \cap W^2 \subseteq G’ rsgi-hb. As such we have a \stackrel{G’ rsgi-hb}{\rightarrow} b \stackrel{G’ mo}{\rightarrow} a, contradicting the assumption that G’ is RSI-consistent.
We proceed by contradiction. Let us assume that there exists \(a, b \in G'.\text{rsi-hb}\). As such we have \(G' \to a \text{ and } b \to G'.\text{mo}\), contradicting the assumption that \(G'\) is RSI-consistent.

Similarly, in cases (4, 5) since \(a, b \in G.W = G'.W\), from Lemma 16 we have either i) \((a, b) \in G'.\text{rsi-hb}\); or ii) \(\exists d \in \xi_a.W\). \(a \to G'.\text{po}\) \(\text{and } d \to (d, b) \in G'.\text{rsi-hb}\). On the other hand, since in case (ii) \(a, d \in G.\xi_a\) \(a \to G'.\text{po}\) \d and \(a, d\) are both write events, we also have \(G'.\text{rsi-hb}\). Moreover, since \(G'.\text{po} \cap W^2 \subseteq G'.\text{rsi-hb}\), we have \((a, d) \in G'.\text{rsi-hb}\). As such we have \(G'.\text{rsi-hb}\) \(d \to G'.\text{rsi-hb}\) \b and from the transitivity of \(G'.\text{rsi-hb}\) in both cases we have \((a, b) \in G'.\text{rsi-hb}\). Consequently, we have \(G' \to b \to a\), contradicting the assumption that \(G'\) is RSI-consistent.

**RTS. part 3**

We proceed by contradiction. Let us assume that there exists \(a, b\) such that \((a, b) \in G.\text{hb}\) and \((b, a) \in G.\text{rb}\). Let \(\text{loc}(a) = \text{loc}(b) = x\). There are then five cases to consider: i) \(\exists (a, b) \in G.\xi_a\) \(\text{and } b \in G.\xi_b\); or ii) \(\exists a, b \in G.\xi_f\); or iii) \(\exists a \in G.\xi_a\) \(b \in G.\xi_b\); or iv) \(\exists a \in G.\xi_a\) and \(b \in G.\xi_b\); or v) \(\exists a \in G.\xi_a\) \(b \in G.\xi_b\) and \(\xi_a \neq \xi_b\).

In case (1) from Lemma 16 we have \((a, b) \in G.\text{po}\). On the other hand since \((b, a) \in G.\text{rb}\), from the construction of \(G\) we know that \((b, a) \in G.\text{po}\). As such, since \(G.\text{po}\) is transitively closed and \(a \to G.\text{po}\) \(b \to G.\text{po}\) \a, we have \((a, a) \in G.\text{po}\), leading to a contradiction as \(G\) is acyclic by construction.

In case (2) from the construction of \(G.\text{rb}\) we know that \((a, b) \in G'.\text{rb}\). On the other hand, from Lemma 16 we have \((a, b) \in G'.\text{rsi-hb}\). As such we have \(G'.\text{rsi-hb}\) \(b \to G'.\text{rb}\) \a, contradicting the assumption that \(G'\) is RSI-consistent.

In case (3) from the construction of \(G.\text{rb}\) we know that \(b = \xi_b.\text{rs}_b \lor b = \xi_b.\text{vs}_b\). Consequently, from Lemma 16 we have \(\{a\} \times G'.\xi_a \subseteq G'.\text{rsi-hb}\). On the other hand, from the construction of \(G.\text{rb}\) we know there exists \(r \in G'.\xi_a\) such that \((r, a) \in G'.\text{rb}\). As such, we have \(G'.\text{rsi-hb}\) \(r \to G'.\text{rb}\) \a, contradicting the assumption that \(G'\) is RSI-consistent.

In case (4) from the construction of \(G.\text{rb}\) we know that \((a, b) \in G'.\text{rb}\). Moreover, since \(a \in G.W\), from Lemma 16 we have either i) \((a, b) \in G'.\text{rsi-hb}\); or ii) \(\exists d \in \xi_a.W\). \(a \to G'.\text{po}\) \(d \land (d, b) \in G'.\text{rsi-hb}\). As such, since \(a, d\) are both write events and \(a \to G'.\text{po}\) \d, we also have \(G'.\text{rsi-hb}\). Moreover, since \(G'.\text{po} \cap W^2 \subseteq G'.\text{rsi-hb}\), we have \((a, d) \in G'.\text{rsi-hb}\). As such we have \(G'.\text{rsi-hb}\) \(d \to G'.\text{rsi-hb}\) \b and from the transitivity of \(G'.\text{rsi-hb}\) in both cases (i, ii) we have \((a, b) \in G'.\text{rsi-hb}\). Consequently, we have \(a \to G'.\text{rb}\) \b, contradicting the assumption that \(G'\) is RSI-consistent.

In case (5) from the construction of \(G.\text{rb}\) we know that \(b = \xi_b.\text{rs}_b \lor b = \xi_b.\text{vs}_b\). Consequently, from Lemma 16 and since \(a\) is a write event, we have either i)
$G'.\mathcal{T}_{\xi_a} \times G'.\mathcal{T}_{\xi_b} \subseteq G'.rsi-hb$; or ii) $\exists d \in \xi_a.W. \ a \xrightarrow{G.poi} d \wedge \{d\} \times G'.\mathcal{T}_{\xi_b} \subseteq G'.rsi-hb$.

As such, since $a, d$ are both write events and $a \xrightarrow{G.poi} d$, we also have $a \xrightarrow{G'.poi} d$. Moreover, since $G'.poi \cap W^2 \subseteq G'.rsi-hb$, we have $(a, d) \in G'.rsi-hb$. As such we have $a \xrightarrow{G'.rsi-hb} d \xrightarrow{G'.rji-hb} b$ and from the transitivity of $G'.rsi-hb$ in both cases (i, ii) we have $\{a\} \times G'.\mathcal{T}_{\xi_b} \subseteq G'.rsi-hb$. On the other hand, from the construction of $G.rb$ we know there exists $r \in G'.\mathcal{T}_{\xi_b}$ such that $(r, a) \in G'.rb$. Consequently, we have $a \xrightarrow{G'.rji-hb} b \xrightarrow{G'.rb} a$, contradicting the assumption that $G'$ is RSI-consistent.
F  Soundness and Completeness of the Lazy RSI Implementation

Given an execution graph \( G \) of the lazy RSI implementation, let us assign a transaction identifier to each transaction executed by the program; and given a transaction \( \xi \). Let \( RS^0_\xi = WS^0_\xi = \emptyset \). Let us write \( G \setminus T \) for those events in \( G \) that do not occur in a transaction. Observe that given a transaction \( \xi \) of the lazy RSI implementation, the trace of \( \xi \), written \( \theta_\xi \), is of the form:

\[
\begin{align*}
FS & \xrightarrow{po|imm} IS \xrightarrow{po|imm} TS \xrightarrow{po|imm} VRs \xrightarrow{po|imm} RUs \xrightarrow{po|imm} PLs \xrightarrow{po|imm} Ws \xrightarrow{po|imm} Us \\
\end{align*}
\]

where:

- \( FS \) denotes the sequence of events failing to obtain the necessary locks (i.e. those iterations that do not succeed in promoting the writer locks) or validate the snapshot;
- \( IS \) denotes the sequence of events initialising the values of \( LS, RS \) and \( WS \) with \( \emptyset \), initialising \( ws \) with \( \emptyset \) and initialising \( s[x] \) with \((\bot, \bot)\) for each location \( x \);
- \( Ts \) denotes the sequence of events corresponding to the execution of \( LT \) and is of the form \( t_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t_k \), where for all \( m \in \{1 \cdots k\} \):

\[
\begin{align*}
t_m = \begin{cases} 
rd(x_m, v_m, RS_{m-1}, WS_{m-1}) & \text{if } \theta_m = R(x_m, v_m) \\
wr(x_m, v_m, RS_{m-1}, WS_{m-1}) & \text{if } \theta_m = W(x_m, v_m) \\
 R(s[x_m], (v'_m, \bot)) & \text{if } \theta_m = R(s[x_m], (v'_m, \bot)) \text{ and for all } m > 0: \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]

\( f_{s_m} \) denotes the sequence of events attempting (but failing) to acquire the read lock on \( x_m \), \( r_{s_m} \triangleq RL(x_1 \cdot m) \), \( u_{rsx_m} \triangleq W(RS, RS_m) \), \( r_{s_m} \triangleq R(x_m, v_0, x_m) \), \( ws_{s_m} \triangleq W(s[x_m], (v'_0, v_0)) \); and for all \( m > 0 \):

\[
RS_{m+1} \triangleq \begin{cases}
RS_m \cup \{x_m\} & \text{if } \theta_m = R(x_m, v_m) \\
RS_m & \text{otherwise}
\end{cases}
\]
and

\[ \text{wr}(x_m, v_m, RS_{m-1}, WS_{m-1}) = \begin{cases} R(s[x_m], (\perp, \perp)) & \text{if } x_m \not\in RS_{m-1} \cup WS_{m-1} \\ \text{pol}_{\text{im}m} \rightarrow f_{sm} & \text{otherwise} \end{cases} \]

\[ w_{ws_{x_m}} = W(WS, WS_m) ; lw_{x_m} = W(s[x_m], (v'_m, v_m)) ; \text{fs}_m \text{ and } rl_{x_m} \text{ are as defined above;} \]

and for all \( m > 0 \):

\[ \text{ws}_{m+1} = \begin{cases} W_{s_m}(x_m, v_m) & \text{if } O_m = W(x_m, v_m) \\ \text{ws}_m & \text{otherwise} \end{cases} \]

Let \( RS_\xi = RS_m, WS_\xi = WS_m, \) and \( ws_\xi = ws_m; \) let \( RS_\xi \cup WS_\xi \) be enumerated as \( \{x_1 \cdots x_i\} \) for some \( i, \) and \( ws_\xi \) be enumerated as \( \{(x_1, v_1) \cdots (x_j, v_j)\} \) for some \( j. \)

- \( VRs \) denotes the sequence of events validating the reads and is of the form \( \text{v}_{x_1} \rightarrow \cdots \rightarrow \text{v}_{x_i} \), where for all \( n \in \{1 \cdots i\} \):

\[ \text{v}_{x_n} = \begin{cases} R(s[x_n], (v_{x_n}^0, -)) & \text{if } x_n \in RS_\xi \\ \text{0} & \text{otherwise} \end{cases} \]

- \( RUs \) denotes the sequence of events releasing the reader locks (when the given location is in the read set only) and is of the form \( \text{ru}_{x_1} \rightarrow \cdots \rightarrow \text{ru}_{x_i} \), where for all \( n \in \{1 \cdots i\} \):

\[ \text{ru}_{x_n} = \begin{cases} \text{RU}(x_n) & \text{if } x_n \not\in WS_\xi \\ \text{0} & \text{otherwise} \end{cases} \]

- \( PLs \) denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form \( \text{pl}_{x_1} \rightarrow \cdots \rightarrow \text{pl}_{x_i} \), where for all \( n \in \{1 \cdots i\} \):

\[ \text{pl}_{x_n} = \begin{cases} \text{PL}(x_n) & \text{if } x_n \in WS_\xi \\ \text{0} & \text{otherwise} \end{cases} \]

- \( Ws \) denotes the sequence of events committing the writes of \( \{(T)\) and is of the form \( c_{x_1, v_1} \rightarrow \cdots \rightarrow c_{x_j, v_j} \), where for all \( n \in \{1 \cdots j\} \): \( c_{x_n, v_n} = W(x_n, v_n) \)

- \( Us \) denotes the sequence of events releasing the locks on the write set, and is of the form \( \text{wu}_{x_1} \rightarrow \cdots \rightarrow \text{wu}_{x_i} \), where for all \( n \in \{1 \cdots i\} \):

\[ \text{wu}_{x_n} = \begin{cases} \text{WU}(x_n) & \text{if } x_n \in WS_\xi \\ \text{0} & \text{otherwise} \end{cases} \]
Given a transaction trace $\theta_\xi$, we write e.g. $\xi.Us$ to refer to its constituent $Us$ sub-trace and write $Us.E$ for the set of events related by po in $Us$. Similarly, we write $\xi.E$ for the set of events related by po in $\theta_\xi$. Note that $G.E = \bigcup_{\xi \in Tx} \xi.E$.

Note that for each transaction $\xi$ and each location $x$, the $\xi.rlx, \xi.rsx, \xi.wr_x, \xi.ru_x$ and $\xi.wu_x$ are uniquely identified when they exist. Let $\xi.wx$ the last (in po order) write to $x$ in $Ws$, when it exists.

For each location $x \in Ws_\xi$, let $fw_x$ denote the maximal write (in po order) logging a write for $x$ in $s[x]$. That is, when $\theta_\xi = t_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t_m$, let $fw_x = \text{wmax}(x, [t_1 \cdots t_m])$, where

$$\text{wmax}(x, L++[t]) \triangleq \begin{cases} lw_x & \text{if } t = \text{wr}(x, -, -, -) \xrightarrow{po} lw_x \xrightarrow{po} \text{wws}_x \\ \text{wmax}(x, L) & \text{otherwise} \end{cases}$$

\section*{F.1 Implementation Soundness}

In order to establish the soundness of our implementation, it suffices to show that given an RA-consistent execution graph of the implementation $G$, we can construct a corresponding RSI-consistent execution graph $G'$ with the same outcome.

Given a transaction $\xi \in Tx$ with $Rs_\xi \cup Ws_\xi = \{x_1 \cdots x_i\}$ and trace $\theta_\xi$ as above with $Ts = t_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t_k$, we construct the corresponding RSI execution trace $\theta'_\xi$ as follows:

$$\theta'_\xi \triangleq t'_1 \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} t'_k$$

where for all $m \in \{1 \cdots k\}$:

$$t'_m = \text{R}(x_m, v_m) \quad \text{when} \quad t_m = \text{rd}(x_m, v_m, S_m) \xrightarrow{po|imm} \text{lr}_{x_m}$$

$$t'_m = \text{W}(x_m, v_m) \quad \text{when} \quad t_m = \text{wr}(x_m, v_m, S_m) \xrightarrow{po|imm} \text{wws}_{x_m} \xrightarrow{po|imm} \cdots$$

such that in the first case the identifier of $t'_m$ is that of $\text{lr}_{x_m}$; and in the second case the identifier of $t'_m$ is that of $\text{wws}_{x_m}$. Note that for each write operation $w$ in $\theta'_\xi$, there exists a matching write operation in $\theta_\xi. Ws$, denoted by $mw(w)$. That is,

$$mw(w) = w' \overset{\text{def}}{=} \exists i. (\theta'_\xi.E \cap W) \mid i = w \land (\theta_\xi.Ws) \mid i = w'$$
We then define:

\[ RF_\xi \triangleq \begin{cases} 
(t', E) \in \mathfrak{T}_\xi & \quad \text{when } a \in \theta_\xi \quad \text{and } \quad a \in G.N.T \\
(w, t') & \quad \text{when } a \in \theta_\xi \\
\text{otherwise} & \quad \text{when } a \notin G.N.T
\end{cases} \]

Similarly, we define:

\[ MO \triangleq \begin{cases} 
\{(w_1, w_2) \mid \exists w_1', w_2'. w_1' = mo(w_1) \land w_2' = mo(w_2) \land (w_1', w_2') \in G.mo\} \\
\cup \{(w, w') \mid w, w' \in G.N.T \land (w, w') \in G.mo\} \\
\cup \{(w, w') \mid w' \in G.N.T \land w' = mo(w) \land (w', w) \in G.mo\} \\
\cup \{(w, w') \mid w' \in G.N.T \land w'' = mo(w) \land (w'', w') \in G.mo\} \\
\end{cases} \]

We are now in a position to demonstrate the soundness of our implementation. Given an RA-consistent execution graph \( G \) of the implementation, we construct an RSI execution graph \( G' \) as follows and demonstrate that rsi-consistent\((G')\) holds.

1. \( G'.E = \bigcup_{\xi \in T_\xi} \theta_\xi \cup G.N.T \), with the \( tx(\cdot) \) function defined as:

\[ tx(a) \triangleq \xi \quad \text{when } a \in \theta_\xi \quad \text{and } a \in G.N.T \]

2. \( G'.po = G.po\mid_{G'.E} \)
3. \( G'.rf = \bigcup_{\xi \in T_\xi} RF_\xi \)
4. \( G'.mo = MO \)
5. \( G'.lo = \emptyset \)

Observe that the events of each \( \theta_\xi \) trace coincides with those of the equivalence class \( T_\xi \) of \( G' \). That is, \( \theta_\xi \cup G.N.T = T_\xi \).

**Lemma 17.** Given an RA-consistent execution graph \( G \) of the implementation and its corresponding RSI execution graph \( G' \) constructed as above, for all \( a, b, \xi_a, \xi_b, x \):

\[ \xi_a \neq \xi_b \land a \in \xi_a \land b \in \xi_b \land E \land \text{loc}(a) = \text{loc}(b) = x \Rightarrow \]

\[ ((a, b) \in G'.rf \Rightarrow \xi_a.wu_a G.bb \xi_b.rl_a) \quad (34) \]

\[ \land ((a, b) \in G'.mo \Rightarrow \xi_a.wu_a G.bb \xi_b.rl_a) \quad (35) \]

\[ \land ((a, b) \in G'.\text{rb} \Rightarrow (x \in WS_{\xi_a} \land \xi_a.wu_a G.bb \xi_b.rl_a) \lor (x \notin WS_{\xi_a} \land \xi_a.ru_a G.bb \xi_b.rl_a)) \quad (36) \]

\[ \land ((a, b) \in G'(mo; rf) \Rightarrow \xi_a.wu_a G.bb \xi_b.rl_a) \quad (37) \]
Proof. Pick an arbitrary RA-consistent execution graph $G$ of the implementation and its corresponding RSI execution graph $G'$ constructed as above. Pick an arbitrary $a, b, \xi_a, \xi_b, x$ such that $\xi_a \neq \xi_b$, $a \in \xi_a.E$, $a \in \xi_a.E$, and $\text{loc}(a) = \text{loc}(b) = x$.

\textbf{RTS. (34)}

Assume $(a, b) \in G'.rf$. From the definition of $G'.rf$ we then know $(\xi_a.w_x, \xi_b.rs_x) \in G'.rf$. On the other hand, from Lemma 2 we know that either i) $x \in \text{WS}_{\xi_a}$ and $\xi_b.w_u_x G^{bb} \xi_a.r_l_x$; or ii) $x \notin \text{WS}_{\xi_a}$ and $\xi_b.ru_x G^{bb} \xi_a.pl_x$; or iii) $\xi_a.w_u_x G^{pp} G^{gg} \xi_b.w_u_x G^{bb} \xi_a.r_l_x$. In case (i) we then have $\xi_a.w_x G^{rf} \xi_b.rs_x G^{po} \xi_b.w_u_x G^{bb} \xi_a.r_l_x \xi_a.w_x$. That is, we have $\xi_a.w_x G^{bb} \xi_a.w_x$, contradicting the assumption that $G$ is RA-consistent. Similarly in case (ii) we have $\xi_a.w_x G^{rf} \xi_b.rs_x G^{po} \xi_b.w_u_x G^{bb} \xi_a.w_x$. That is, we have $\xi_a.w_x G^{bb} \xi_a.w_x$, contradicting the assumption that $G$ is RA-consistent. In case (iii) the desired result holds trivially.

\textbf{RTS. (35)}

Assume $(a, b) \in G'.mo$. From the definition of $G'.mo$ we then know there exist $w_1 \in \xi_a.Ws$ and $w_2 \in \xi_a.Ws$ such that $(w_1, w_2) \in G'.mo$ and $\text{loc}(w_1) = \text{loc}(w_2) = x$. On the other hand, from Lemma 2 we know that either i) $\xi_b.w_u_x G^{bb} \xi_a.pl_x$; or ii) $\xi_a.w_u_x G^{bb} \xi_b.r_l_x$. In case (i) we then have $w_1 G^{mo} w_2 G^{po} \xi_b.w_u_x G^{bb} \xi_a.r_l_x \xi_a.w_x$. That is, we have $w_1 G^{mo} w_2 G^{bb} \xi_a.w_x$, contradicting the assumption that $G$ is RA-consistent. In case (ii) the desired result holds trivially.

\textbf{RTS. (36)}

Assume $(a, b) \in G'.rb$. From the definition of $G'.rb$ we then know that there exist $w \in \xi_b.Ws$ such that $\text{loc}(w) = x$ and $(\xi_a.rs_x, w) \in G'.rb$. From Lemma 2 we then know that either i) $\xi_b.w_u_x G^{bb} \xi_a.r_l_x$; or ii) $x \notin \text{WS}_{\xi_a}$ and $\xi_a.ru_x G^{bb} \xi_a.pl_x$; or iii) $x \in \text{WS}_{\xi_a}$ and $\xi_a.w_u_x G^{bb} \xi_a.r_l_x$. In case (i) we then have $w G^{po} \xi_a.w_u_x G^{bb} \xi_a.r_l_x \xi_a.r_l_x G^{bb} \xi_a.rs_x G^{rb} w$. That is, we have $w G^{bb} \xi_a.rs_x G^{rb} w$, contradicting the assumption that $G$ is RA-consistent. In cases (ii-iii) the desired result holds trivially.

\textbf{Lemma 18.} For all RA-consistent execution graphs $G$ of the implementation and their counterpart RSI execution graphs $G'$ constructed as above and $S \triangleq$
\(G'.(\text{rsi-po} \cup \text{rsi-rf} \cup \text{mo}_T)^+\):

\[
\forall \xi_a, \xi_b, a, b. \forall a, b.
(a, b) \in S \Rightarrow (a, b) \in G'.N^T \land (a, b) \in G.hb)
\]

\[
\lor (a \in G'.T_{\xi_a} \land b \in G'.T_{\xi_b} \land \xi_a = \xi_b \land (a, b) \in G.po)
\]

\[
\lor (a \in G'.T_{\xi_a} \land b \in G'.T_{\xi_b} \land \xi_a \neq \xi_b)
\]

\[
\lor \exists d \in G'.T_{\xi_a}. \forall c \in G'.T_{\xi_c}. (c, d) \in G.hb
\]

\[
\land \exists e \in \xi_a.Ws. (e, d) \in G.hb
\]

\[
\lor \exists e \in \xi_a.Ws. (e, b) \in G.hb
\]

\[
\lor \exists e \in \xi_a.E. (c, b) \in G.hb
\]

\[
\lor \exists e \in \xi_a.E. (c, b) \in G.hb
\]

Proof. Let \(S_0 = G'.(\text{rsi-po} \cup \text{rsi-rf} \cup \text{mo}_T)\), and \(S_{n+1} = S_0; S_n\), for all \(n \geq 0\). It is straightforward to demonstrate that \(S = \bigcup_{i \in \mathbb{N}} S_i\). We thus demonstrate instead that:

\[
\forall i \in \mathbb{N}. \forall \xi_a, \xi_b, a, b. \forall a, b.
(a, b) \in S_i \Rightarrow (a, b) \in G'.N^T \land (a, b) \in G.hb)
\]

\[
\lor (a \in G'.T_{\xi_a} \land b \in G'.T_{\xi_b} \land \xi_a = \xi_b \land (a, b) \in G.po)
\]

\[
\lor (a \in G'.T_{\xi_a} \land b \in G'.T_{\xi_b} \land \xi_a \neq \xi_b)
\]

\[
\lor \exists d \in G'.T_{\xi_a}. \forall c \in G'.T_{\xi_c}. (c, d) \in G.hb
\]

\[
\land \exists e \in \xi_a.Ws. (e, d) \in G.hb
\]

\[
\lor \exists e \in \xi_a.Ws. (e, b) \in G.hb
\]

\[
\lor \exists e \in \xi_a.E. (c, b) \in G.hb
\]

\[
\lor \exists e \in \xi_a.E. (c, b) \in G.hb
\]

We proceed by induction on \(i\).

**Base case** \(i = 0\)

Pick arbitrary \(\xi_a, \xi_b, a, b\) such that \(\xi_a \neq \xi_b\) and \((a, b) \in S_0\). There are now four cases to consider: \(A\) \(a, b \in G'.T_{\xi_a}; \ b \in G'.T_{\xi_b}\); or \(B\) \(a, b \in G'.N^T; \ c \in G'.N^T; \ b \in G'.T_{\xi_b}\); or \(C\) \(a, b \in G'.N^T; \ b \in G'.T_{\xi_b}\); or \(D\) \(a \in G'.N^T; \ b \in G'.T_{\xi_b}\).

In case \(A\) there are three additional cases to consider: 1) \((a, b) \in G'.\text{rsi-po}\); or 2) \((a, b) \in G'.\text{rsi-rf}\); or 2) \((a, b) \in G'.\text{mo}_T\).

In case \(A.1\), pick an arbitrary \(c \in G'.T_{\xi_a}\). From the definition of \(G'.\text{po}_T\) we have \((c, b) \in G.po \subseteq G.hb\), as required. Now assume that \(a \in W\). From the definition of \(G'.\text{po}\) we then have \((w(a), d) \in G.po \subseteq G.hb\), as required.

In case \(A.2\), we then know there exists \(w \in G'.T_{\xi_a}\) and \(r \in G'.T_{\xi_b}\) such that \((w, r) \in G'.\text{rf}\). Let \(1\text{oc}(w) = 1\text{oc}(r) = x\). From Lemma \ref{lem17} we then have \(\xi_a.wu_x \rightarrow G.hb\). Pick an arbitrary \(c \in G'.T_{\xi_a}\). As such we have...
In case (A.3), we then know there exists \( w \in G' \cdot T_{\xi_a} \) and \( w' \in G' \cdot T_{\xi_b} \) such that \( (w, w') \in G' \cdot \rho_0 \). Let \( \text{loc}(w) = \text{loc}(w') = x \). From Lemma 17 we then have \( \xi_a.wu_x \rightarrow_{G' \cdot \rho_0} \xi_b.rl_x \). Pick an arbitrary \( c \in G' \cdot T_{\xi_a} \). As we also have \( \xi_b.rl_x \rightarrow_{G' \cdot \rho_0} w_2 \), \( w_1 \rightarrow_{G' \cdot \rho_0} \xi_a.wu_x \), and \( c \rightarrow_{G' \cdot \rho_0} \xi_a.wu_x \), we then have \( (c, w_2) \in G' \cdot \rho_0 \) and \( (w_1, w_2) \in G' \cdot \rho_0 \), as required. Now assume that \( a \in W \). We then have \( (mw(a), \xi_a.wu_x) \in G' \cdot \rho_0 \). As such, from the transitivity of \( G' \cdot \rho_0 \) we also have \( (mw(a), \xi_b.rs_x) \in G' \cdot \rho_0 \), as required.

In case (B.1) we then have \( (a, b) \in G' \cdot \rho_0 \) and from the definition of \( G' \cdot \rho_0 \) we have \( (a, b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. In case (B.2) we then have \( (a, b) \in G' \cdot \rho_0 \) and from the definition of \( G' \cdot \rho_0 \) we have \( (a, b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. Case (B.3) does not apply as \( a, b \in G' \cdot \rho_0 \).

In case (C) there are three additional cases to consider: 1) \( (a, b) \in G' \cdot \rho_0 \); or 2) \( (a, b) \in G' \cdot \rho_0 \); or 3) \( (a, b) \in G' \cdot \rho_0 \).

In case (C.1) we then have \( (a, b) \in G' \cdot \rho_0 \) and from the definition of \( G' \cdot \rho_0 \) we have \( (a, b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. In case (C.2) we know there exists \( r \in G' \cdot T_{\xi_a} \) such that \( (a, r) \in G' \cdot \rho_0 \). Let \( \text{loc}(a) = \text{loc}(r) = x \). From the definition of \( G' \cdot \rho_0 \) we then have \( (a, \xi_b.rs_x) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. Case (C.3) does not apply as \( a \in G' \cdot \rho_0 \).

In case (D) there are three additional cases to consider: 1) \( (a, b) \in G' \cdot \rho_0 \); or 2) \( (a, b) \in G' \cdot \rho_0 \); or 3) \( (a, b) \in G' \cdot \rho_0 \).

In case (D.1) we then have \( (a, b) \in G' \cdot \rho_0 \). Pick an arbitrary \( c \in G' \cdot T_{\xi_a} \). From the definition of \( G' \cdot \rho_0 \) we have \( (c, b) \in G' \cdot \rho_0 \). As such, from the definition of \( G' \cdot \rho_0 \) we have \( (c, b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. Now assume that \( a \in W \). We then have \( (mw(a), b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required.

In case (D.2) we then have \( (a, b) \in G' \cdot \rho_0 \) and from the definition of \( G' \cdot \rho_0 \) we know that \( mw(a) \in G' \cdot W \) and \( (mw(a), b) \in G' \cdot \rho_0 \subseteq G' \cdot \rho_0 \), as required. Pick an arbitrary \( c \in G' \cdot T_{\xi_a} \). We then know \( (c, mw(a)) \in G' \cdot \rho_0 \). We then have \( c \rightarrow_{G' \cdot \rho_0} mw(a) \rightarrow_{G' \cdot \rho_0} b \). As such, from the definition of \( G' \cdot \rho_0 \) we have \( (c, b) \in G' \cdot \rho_0 \) as required. Case (D.3) does not apply as \( b \in G' \cdot \rho_0 \).
We proceed by contradiction. Let us assume \( \neg G \) is RA-consistent. G

\[ a \in G', T_{\xi_a} \land b \in G', T_{\xi_b} \land \xi_a = \xi_b \land (a, b) \in G, po \]

\[ \exists e_d \in G', T_{\xi_a}, \forall c \in G', T_{\xi_c}, (c, d) \in G, hb \]

\[ \wedge (\exists e \in \xi_a, Ws, (e, d) \in G, hb) \]

\[ \forall c \in \xi_a, E, (c, d) \in G, hb \]

(I.H.)

Pick arbitrary \( \xi_a, \xi_b, a \in G', T_{\xi_a}, b \in G', T_{\xi_b} \) such that \( (a, b) \in S_i \). From the definition of \( S_i \) we then know there exist \( e, \xi_e \) such that \( e \in G', T_{\xi_e}, (e, c) \in S_0 \) and \( (e, b) \in S_n \). The desired result then follows from the inductive hypothesis and case analysis on \( a, b \) and \( c \).

**Theorem 13 (Soundness).** For all execution graphs \( G \) of the implementation and their counterpart RSI execution graphs \( G' \) constructed as above,

\[
\text{RA-consistent}(G) \Rightarrow \text{rsi-consistent}(G')
\]

**Proof.** Pick an arbitrary execution graph \( G \) of the implementation such that RA-consistent\( (G) \) holds, and its associated RSI execution graph \( G' \) constructed as described above.

**RTS. irreflexive\( (G', \text{rsi-hb}) \)**

We proceed by contradiction. Let us assume \( \neg \text{irreflexive}(G', \text{rsi-hb}) \) Let \( S = G', (\text{rsi-po} \cup \text{rsi-rf} \cup \text{mo})^+ \). There are now two cases to consider: either there is an \( \text{rsi-hb} \) cycle without a \( \text{si-rb} \) edge; or there is a cycle with one or more \( \text{si-rb} \) edges. That is, either 1) there exists a such that \( (a, a) \in S \); or 2) there exist \( a_1, b_1, \ldots, a_n, b_n \) such that \( a_1 \xrightarrow{G', \text{si-rb}} b_1 \xrightarrow{S} a_2 \xrightarrow{G', \text{si-rb}} b_2 \xrightarrow{S} \cdots \xrightarrow{S} a_n \xrightarrow{G', \text{si-rb}} b_n \xrightarrow{S} a_1 \).

In case (1) we then know that either i) there exists \( \xi \) such that \( a \in G', T_{\xi_a} \); or ii) \( a \in G', NT \). In case (1.i) from Lemma 18 we then have \( (a, a) \in G, po \), contradicting the assumption that \( G \) is RA-consistent. Similarly, in case (1.ii) from Lemma 18 we know that \( (a, a) \in G, hb \), contradicting the assumption that \( G \) is RA-consistent.

In case (2), for an arbitrary \( i \in \{1 \cdots n\} \), let \( j = i+1 \) when \( i \neq n \); and \( j = 1 \) when \( i = n \). As \( a_i \xrightarrow{G', \text{si-rb}} b_i \), we know there exists \( \xi_{a_i}, \xi_{b_i} \) such that \( a_i \in T_{\xi_{a_i}}, b_i \in T_{\xi_{b_i}}, \) and that there exist \( r_i \in T_{\xi_{a_i}} \cap R_E \) and \( w_i \in T_{\xi_{b_i}} \cap W \).
such that \((r_i, w_i) \in G'.\hb\). Let \(\text{loc}(r_i) = \text{loc}(w_i) = x_i\). From Lemma 17 we then know that either i) \(\xi_{a_i}.r_{u_{x_i}} \xrightarrow{G.\hb} \xi_{b_i}.p_{l_{x_i}}\); or ii) \(\xi_{a_i}.w_{u_{x_i}} \xrightarrow{G.\hb} \xi_{b_i}.r_{l_{x_i}}\). Note that for all \(w' \in \xi_{b_i}.\Ws\), we know \(\xi_{b_i}.r_{l_{x_i}} \rightarrow w'\) and \(\xi_{b_i}.p_{l_{x_i}} \rightarrow w'\). As such, as \((b_i, a_j) \in S\), from Lemma 18 and since \(G.\hb\) is transitively closed, we know there exists \(d_j \in T_{\xi_{a_j}}\) such that either \(\xi_{a_i}.r_{u_{x_i}} \xrightarrow{G.\hb} d_j\), or \(\xi_{a_i}.w_{u_{x_i}} \xrightarrow{G.\hb} d_j\). That is, \(\xi_{a_i}.u_{x_i} \xrightarrow{G.\hb} d_j\), where either \(u_{x_i} = r_{u_{x_i}}\), or \(u_{x_i} = w_{u_{x_i}}\). On the other hand, observe that for all \(d_i \in T_{\xi_{a_i}}\) we have \(d_i \xrightarrow{G.\po} \xi_{a_i}.u_{x_i}\), i.e. \(d_i \xrightarrow{G.\hb} \xi_{a_i}.u_{x_i}\). As such, we have \(d_j \xrightarrow{G.\hb} \xi_{a_j}.u_{x_j}\). As we also have \(\xi_{a_i}.u_{x_i} \xrightarrow{G.\hb} d_j\) and \(G.\hb\) is transitively closed, we have \(\xi_{a_i}.u_{x_i} \xrightarrow{G.\hb} \xi_{a_j}.u_{x_j}\). We then have \(\xi_{a_i}.u_{x_i} \xrightarrow{G.\hb} \xi_{a_j}.u_{x_j}\) \(\xrightarrow{G.\hb} \xi_{a_2}.u_{x_2}\) \(\xrightarrow{G.\hb} \ldots \xrightarrow{G.\hb} \xi_{a_{n-1}}.u_{x_{n-1}}\). That is, \(\xi_{a_1}.u_{x_1} \xrightarrow{G.\hb} \xi_{a_n}.u_{x_n}\), contradicting the assumption that \(G\) is RA-consistent.

**RTS.** \(r_{f_1} \cup m_{o_1} \cup r_{b_1} \subseteq p_o\)

Follows immediately from the construction of \(G'\).

**RTS. irreflexive** \((G'.(\rshi; \mo))\)

We proceed by contradiction. Let us assume \(\neg\text{irreflexive}(G'.(\rshi; \mo))\).

That is, there exists \(a, b\) such that \((a, b) \in \rshi\) and \((b, a) \in \mo\). There are now five cases to consider: 1) there exists \(\xi\) such that \(a, b \in T_\xi\); or 2) there exists \(\xi, \xi_b\) such that \(\xi \neq \xi_b\), \(a \in T_\xi\) and \(b \in T_{\xi_b}\); or 3) there exists \(\xi\) such that \(a \in T_\xi\) and \(b \in G'.N.T\); or 4) there exists \(\xi\) such that \(b \in T_\xi\) and \(a \in G'.N.T\); or 5) \(a, b \in G'.N.T\).

In case 1) we then have \((b, a) \in m_{o_1} \subseteq p_{o_1}\) (from the proof of the previous part) and since \(a, b\) are both write events, we have \((b, a) \subseteq \rshi\). We then have \(a \xrightarrow{\rshi} b \xrightarrow{\rshi} a\), contradicting our proof above that \(\rshi\) is irreflexive.

In case 2) we then have \((b, a) \in m_{o_T} \subseteq \rshi\). We then have \(a \xrightarrow{\rshi} b \xrightarrow{\rshi} a\), contradicting our proof above that \(\rshi\) is irreflexive.

In case 3) from Lemma 18 we know \((m_{w}(a), b) \in G.\hb\). From the definition of \(G'.\mo\) we also have \((b, m_{w}(a)) \in G.\mo\). We then have \(m_{w}(a) \xrightarrow{G.\hb} b \xrightarrow{G.\mo} m_{w}(a)\), contradicting the assumption that \(G\) is RA-consistent.

In case 4) from Lemma 18 we know there exists \(d \in T_\xi\) such that \((a, d) \in G.\hb\). From the definition of \(G'.\mo\) we also have \((m_{w}(b), a) \in G.\mo\). Moreover, from the construction of \(G'\) we know \((d, m_{w}(b)) \in G.\po\) We then have \(a \xrightarrow{G.\hb} d \xrightarrow{G.\po} m_{w}(b) \xrightarrow{G.\mo} a\), contradicting the assumption that \(G\) is RA-consistent.

In case 5) from Lemma 18 we know \((a, b) \in G.\hb\). From the definition of \(G'.\mo\) we also have \((b, a) \in G.\mo\). We then have \(a \xrightarrow{G.\hb} b \xrightarrow{G.\mo} a\), contradicting the assumption that \(G\) is RA-consistent.

**RTS. irreflexive** \((G'.(\rshi; \rhr))\)

We proceed by contradiction. Let us assume \(\neg\text{irreflexive}(G'.(\rshi; \rhr))\).

That is, there exists \(a, b\) such that \((a, b) \in \rshi\) and \((b, a) \in \rhr\). There are now
five cases to consider: 1) there exists \( \xi \) such that \( a, b \in T_\xi \); or 2) there exists \( \xi_a, \xi_b \) such that \( \xi_a \neq \xi_b, a \in T_\xi \) and \( b \in T_\xi \); or 3) there exists \( \xi \) such that \( a \in T_\xi \) and \( b \in G'N_T \); or 4) there exists \( \xi \) such that \( b \in T_\xi \) and \( a \in G'N_T \); or 5) \( a, b \in G'N_T \).

In case (1) we then have \( (b, a) \in rb \subseteq po \) (from the proof of the earlier part) and thus \( (b, a) \in G.po \). Moreover, from Lemma 18 we know \( (a, b) \in G.po \) We then have \( a \longrightarrow b \longrightarrow a \), contradicting the assumption that \( G \) is RA-consistent.

In case (2) we then have \( (b, a) \in si-rb \cup mo \subseteq rsi-hb \). We then have \( a \rightarrow b \rightarrow a \), contradicting our proof above that rsi-hb is irreflexive.

In case (3) from Lemma 18 we know \( (mw(a), b) \in G.hb \). From the definition of \( G'.rb \) we also have \( (b, mw(a)) \in G.rb \). We then have \( mw(a) \longrightarrow b \longrightarrow rb \longrightarrow mw(a) \), contradicting the assumption that \( G \) is RA-consistent.

In case (4) from Lemma 18 we know there exists \( d \in T_\xi \) such that \( (a, d) \in G.hb \). Let \( loc(a) = loc(b) = x \). From the definition of \( G'.rb \) and our race-freedom stipulation of non-transactional writes with the same transaction we also have \( (\xi_b, vr_x, a) \in G.rb \). Moreover, from the construction of \( G' \) we know \( (d, \xi_0, vr_x, b) \in G.po \) We then have \( a \rightarrow d \longrightarrow b \longrightarrow a \), contradicting the assumption that \( G \) is RA-consistent.

In case (5) from Lemma 18 we know \( (a, b) \in G.hb \). From the definition of \( G'.rb \) we also have \( (b, a) \in G.rb \). We then have \( a \rightarrow b \rightarrow a \), contradicting the assumption that \( G \) is RA-consistent.

F.2 Implementation Completeness

In order to establish the completeness of our implementation, it suffices to show that given an RSI-consistent execution graph \( G' = (E, po, rf, mo, lo) \), we can construct a corresponding RA-consistent execution graph \( G \) of the implementation.

Note that the execution trace for each transaction \( T_\xi \in G'.T/st \) of is of the form \( \theta_\xi' = t_1' \longrightarrow po \longrightarrow \cdots \longrightarrow po \longrightarrow t_k' \) for some \( k \), where each \( t_j' \) is a read or write event. As such, we have \( G'.T = \bigcup_{T_\xi \in G'.T/st} T_\xi = \theta_\xi' \). For each transaction \( \xi \), we construct the implementation trace \( \theta_\xi \) as follows:

\[
Fs \longrightarrow po \longrightarrow Is \longrightarrow po \longrightarrow Ts \longrightarrow po \longrightarrow VRs \longrightarrow po \longrightarrow RUs \longrightarrow po \longrightarrow PLs \longrightarrow po \longrightarrow Ws \longrightarrow po \longrightarrow Us
\]

where:

- \( Fs \) denotes the sequence of events failing to obtain the necessary locks (i.e. those iterations that do not succeed in promoting the writer locks) or validate the snapshot;
- \( Is \) denotes the sequence of events initialising the values of \( LS, RS \) and \( WS \) with \( \emptyset \), initialising \( ws \) with [] and initialising \( s[x] \) with \( (\perp, \perp) \) for each location \( x \);
- $Ts$ denotes the sequence of events corresponding to the execution of $T$, and is of the form $t_1^{po \downarrow} \cdots t_k^{po \downarrow}$, where for all $m \in \{1 \cdots k\}$:

$$
t_m = \begin{cases} 
rd(x_m, v_m, RS_{m-1}, WS_{m-1})^{po \downarrow} \rightarrow h_{x_m} & \text{if } O_m = R(x_m, v_m) \\
wq(x_m, v_m, RS_{m-1}, WS_{m-1})^{po \downarrow} \rightarrow w_{x_m} & \text{if } O_m = \bar{w}(x_m, v_m)
\end{cases}
$$

where $O_m$ denotes the $m$th event in the trace of $T$; $br_{x_m} \triangleq r(s[x_m], (-, v_m))$;

$$
rd(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} 
R(s[x_m], (\bot, \bot)) & \text{if } x_m \not\in RS_{m-1} \cup WS_{m-1} \\
po \rightarrow fr_{x_m} & \text{if } O_m = R(x_m, -) \\
po \rightarrow rl_{x_m} & \text{otherwise}
\end{cases}
$$

$fs_m$ denotes the sequence of events attempting (but failing) to acquire the read lock on $x_m$, $rl_{x_m} \triangleq RL(x_m)$, $wrs_{x_m} \triangleq w(RS, RS_m)$, $rs_{x_m} \triangleq R(x_m, x_0^0)$, $ws_{x_m} \triangleq w(s[x_m], (v_0^0, v_0^0))$; and for all $m > 0$:

$$
RS_{m+1} \triangleq \begin{cases} 
RS_m \cup \{x_m\} & \text{if } O_m = R(x_m, -) \\
RS_m & \text{otherwise}
\end{cases}
$$

and

$$
wq(x_m, v_m, RS_{m-1}, WS_{m-1}) \triangleq \begin{cases} 
R(s[x_m], (\bot, \bot)) & \text{if } x_m \not\in RS_{m-1} \cup WS_{m-1} \\
po \rightarrow fr_{x_m} & \text{if } O_m = \bar{w}(x_m, -) \\
po \rightarrow rl_{x_m} & \text{otherwise}
\end{cases}
$$

$wrs_{x_m} \triangleq w(WS, WS_m)$; $lw_{x_m} \triangleq w(s[x_m], (v_0', v_m))$; $fs_m$ and $rl_{x_m}$ are as defined above; and for all $m > 0$:

$$
WS_{m+1} \triangleq \begin{cases} 
WS_m \cup \{x_m\} & \text{if } O_m = \bar{w}(x_m, -) \\
WS_m & \text{otherwise}
\end{cases}
$$

$$
w_{x_m+1} \triangleq \begin{cases} 
w_{x_m} + [(x_m, v_m)] & \text{if } O_m = \bar{w}(x_m, v_m) \\
w_{x_m} & \text{otherwise}
\end{cases}
$$

Let $RS_\xi = RS_m$, $WS_\xi = WS_m$, and $ws_\xi = ws_m$; let $RS_\xi \cup WS_\xi$ be enumerated as $\{x_1 \cdots x_i\}$ for some $i$, and $ws_\xi$ be enumerated as $\{(x_1, v_1) \cdots (x_j, v_j)\}$ for some $j$. 
VRs denotes the sequence of events validating the reads and is of the form
\[ v_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} v_{x_i}, \]
where for all \( n \in \{1 \cdots i\} \):
\[ v_{x_n} = \begin{cases} 
R(s[x_n], (v^0_{x_n}, -)) \xrightarrow{po|imm} vr_{x_n} = R(x_n, v^0_{x_n}) & \text{if } x_n \in RS_x \\
\emptyset & \text{otherwise}
\end{cases} \]

RU\text{s} denotes the sequence of events releasing the reader locks (when the given location is in the read set only) and is of the form
\[ ru_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} ru_{x_i}, \]
where for all \( n \in \{1 \cdots i\} \):
\[ ru_{x_n} = \begin{cases} 
RU(x_{1n}) & \text{if } x_n \notin WS_x \\
\emptyset & \text{otherwise}
\end{cases} \]

PL\text{s} denotes the sequence of events promoting the reader locks to writer ones (when the given location is in the write set), and is of the form
\[ pl_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} pl_{x_i}, \]
where for all \( n \in \{1 \cdots i\} \):
\[ pl_{x_n} = \begin{cases} 
PL(x_{1n}) & \text{if } x_n \in WS_x \\
\emptyset & \text{otherwise}
\end{cases} \]

Ws denotes the sequence of events committing the writes of \((T)\) and is of the form
\[ c_{x_1, v_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} c_{x_j, v_j}, \]
where for all \( n \in \{1 \cdots j\} \):
\[ c_{x_n, v_n} = W(x_n, v_n) \]

Us denotes the sequence of events releasing the locks on the write set, and is of the form
\[ wu_{x_1} \xrightarrow{po|imm} \cdots \xrightarrow{po|imm} wu_{x_i}, \]
where for all \( n \in \{1 \cdots i\} \):
\[ wu_{x_n} = \begin{cases} 
WU(x_{1n}) & \text{if } x_n \in WS_x \\
\emptyset & \text{otherwise}
\end{cases} \]

Given a transaction trace \( \theta_\xi \), we write e.g. \( \xi.Us \) to refer to its constituent Us sub-trace and write \( Us.E \) for the set of events related by po in Us. Similarly, we write \( \xi.E \) for the set of events related by po in \( \theta_\xi \). Note that \( G.E = \bigcup_{\xi \in Tx} \xi.E \).

Note that for each transaction \( \xi \) and each location \( x \), the \( \xi.rl_x, \xi.rs_x, \xi.vr_x, \xi.ru_x, \xi.pl_x \) and \( \xi.wu_x \) are uniquely identified when they exist. Let \( \xi.w_x \) the last (in po order) write to \( x \) in Ws, when it exists.

Note that for each write operation \( w \) in \( \theta'_\xi \), there exists a matching write operation in \( \theta_\xi.Ws \), denoted by \( mw(w) \) is a one-to-one function such that: That is,
\[ mw(w) = w' \iff \exists i. (\theta'_\xi.E \cap W)[i] = w \land (\theta_\xi.Ws)[i] = w' \]
We then define:

\[
RF \triangleq \begin{cases} 
(w, r) \mid & r \in G'.N' \wedge (w, r) \in G'.rf \\
\lor & (w \in G.N' \wedge (w, r) \in G'.rf) \\
\lor & (w, \xi, rs_z) \mid (w \in G.N' \wedge (w, \xi, rs_z) \in G'.rf) \\
\lor & (w, \xi, vr_k) \mid (w \in G.N' \wedge (w, \xi, vr_k) \in G'.rf)
\end{cases}
\]

Similarly, we define:

\[
MO \triangleq \begin{cases} 
(w_1, w_2) \mid & \exists w_1', w_2'. w_1 = w(u'_1) \wedge w_2 = w(u_2') \wedge (w_1', w_2') \in G'.mo \\
\lor & (w, w') \mid w, w' \in G.N' \wedge (w, w') \in G'.mo \\
\lor & (w, w') \mid w' \in G.N' \wedge \exists w''. w = w(u''_1) \wedge (w'', w') \in G'.mo \\
\lor & (w', w) \mid w' \in G.N' \wedge \exists w''. w = w(u''_1) \wedge (w', w'') \in G'.mo
\end{cases}
\]

For each location \( x \) we then define:

\[
LO_x \triangleq \begin{cases} 
(\xi, rl_x, \xi, pl_x), (\xi, rl_x, \xi, wt_x), (\xi, pl_x, \xi, wt_x) \mid G'.T_x \cap W_x \neq \emptyset \\
\lor & \begin{cases} 
(\xi, rl_x, \xi', pl_x), (\xi, rl_x, \xi', wt_x), \\
(\xi, rl_x, \xi', wt_x), (\xi, rl_x, \xi', pl_x), (\xi, pl_x, \xi, wt_x), \\
(\xi, wt_x, \xi', rl_x), (\xi, wt_x, \xi', rl_x), (\xi, wt_x, \xi', tw_x)
\end{cases} \\
\lor & \begin{cases} 
(\xi, rl_x, \xi', pl_x), (\xi, rl_x, \xi', tw_x), \\
(\xi, pl_x, \xi', rl_x), (\xi, pl_x, \xi', tw_x), (\xi, pl_x, \xi, tw_x), \\
(\xi, tw_x, \xi', rl_x), (\xi, tw_x, \xi', rl_x), (\xi, tw_x, \xi', tw_x)
\end{cases}
\end{cases}
\]

Note that each \( LO_x \) satisfies the conditions in Def. 4.

We are now in a position to demonstrate the completeness of our implementation.
Given an RSI-consistent execution graph \( G' \), we construct an execution graph \( G \) of the implementation as follows and demonstrate that \( RA\text{-consistent}(G) \) holds.

- \( G.E = \bigcup_{T \in G'.T} \theta_T.E \cup G'.N' \). Observe that \( G'.E \subseteq G.E \).
- \( G.po \) is defined as \( G'.po \) extended by the po for the additional events of \( G \), given by each \( \theta_T \) trace defined above. Note that \( G.po \) does not introduce additional orderings between events of \( G'.E \). That is, \( \forall a, b \in G'.E \), \( (a, b) \in G'.po \) then \( (a, b) \in G.po \).
- \( G.rf = RF \).
- \( G.mo = MO \).
- \( G.lo = \bigcup_{x \in \text{Locs}} LO_x \), with \( LO_x \) as defined above.
Notation. In what follows, given an RSI implementation graph $G$ as constructed above we write $G,\mathcal{N}T$ for the non-transactional events of $G$, i.e. $G,\mathcal{N}T \triangleq \{a \mid \neg \exists \xi. \ a \in G.E_\xi\}$. Moreover, as before, given a relation $r \subseteq G.E \times G.E$, we override the $r_1$ notation and write $r_1$ for $\{ (a, b) \in r \mid \exists \xi. \ a, b \in \theta_\xi.E \}$.

Lemma 19. Given an RSI-consistent execution graph $G'$ and its corresponding implementation graph $G$ constructed as above, for all $a, b, \xi_a, \xi_b$:

$$(a, b) \in G.h_b \Rightarrow (\exists \xi, a, b \in G.E_\xi \Rightarrow (a, b) \in G.po) \land (\exists a, b \in G.E_\xi \Rightarrow \exists A, B, \emptyset \subseteq A \times B \subseteq G'.rsi-hb \land (a \in G,\mathcal{N}T \Rightarrow A=\{a\}) \land (b \in G,\mathcal{N}T \Rightarrow B=\{b\}) \land [a \in G.E_\xi \Rightarrow A=G'.T_\xi \land (stg(a, \xi_a) \leq 2 \land \forall d \in \xi_a.W. a \xrightarrow{\mathcal{N}T} d \land A = \{\mathcal{M}^{-1}(d)\})] \land [b \in G.E_\xi \Rightarrow B=G'.T_\xi \land (stg(b, \xi_b) \geq 3 \land B=G'.T_\xi \land W)] \big)$$

where

$$stg(a, \xi_a) \triangleq \begin{cases} 1 & \text{if } a \notin \xi_a.RUs_{\xi_a} \cup \xi_a.PLs_{\xi_a} \cup \xi_a.Ws_{\xi_a} \cup \xi_a.Us_{\xi_a} \\ 2 & \text{if } a \in \xi_a.RUs_{\xi_a} \\ 3 & \text{if } a \in \xi_a.PLs_{\xi_a} \\ 4 & \text{otherwise} \end{cases}$$

Proof. Pick an arbitrary RSI-consistent execution graph $G'$ and its corresponding implementation graph $G$ constructed as above. Let $h_b_0 \triangleq G.(po \cup rf \cup lo)$ and $h_b_{n+1} \triangleq h_b_n; h_b_n$, for all $n \in \mathbb{N}$. It is then straightforward to demonstrate that $G.h_b = \bigcup_{n \in \mathbb{N}} h_b_n$. We proceed by induction on $n$.

$$\forall n \in \mathbb{N}. \forall a, b, \xi_a, \xi_b. \ (a, b) \in G.h_b_n \Rightarrow (\exists \xi, a, b \in G.E_\xi \Rightarrow (a, b) \in G.po) \land (\exists a, b \in G.E_\xi \Rightarrow \exists A, B, \emptyset \subseteq A \times B \subseteq G'.rsi-hb \land (a \in G,\mathcal{N}T \Rightarrow A=\{a\}) \land (b \in G,\mathcal{N}T \Rightarrow B=\{b\}) \land [a \in G.E_\xi \Rightarrow A=G'.T_\xi \land (stg(a, \xi_a) \leq 2 \land \forall d \in \xi_a.W. a \xrightarrow{\mathcal{N}T} d \land A = \{\mathcal{M}^{-1}(d)\})] \land [b \in G.E_\xi \Rightarrow B=G'.T_\xi \land (stg(b, \xi_b) \geq 3 \land B=G'.T_\xi \land W)] \big)$$
Base case $n = 0$

There are three cases to consider: 1) $(a, b) \in G.p_0$; or 2) $(a, b) \in G.r_0$; or 3) $(a, b) \in G.l_0$.

In case (1) there are five cases to consider: a) $\exists \xi, (a, b) \in G.E_{\xi}$; or b) $a, b \in G.N'_{\xi}$; or c) $a \in G.N'_{\xi} \text{ and } b \in G.E_{\xi}$; or d) $a \in G.E_{\xi}$ and $b \in G.N'_{\xi}$; or e) $a \in G.E_{\xi}$, $b \in G.E_{\xi}$, and $\xi_a \neq \xi_b$. In case (1.a) we then have $(a, b) \in G.p_0$, as required. In case (1.b) from the construction of $G.p_0$ we have $(a, b) \in G.r_0$, as required. In case (1.c) from the construction of $G.p_0$ we have $\{\{a\} \times G'.T_{\xi_a}\} \subseteq G'.r_0$, as required. In case (1.d) from the construction of $G.p_0$ we have $(G'.T_{\xi_a} \times \{b\}) \subseteq G'.r_0$, as required. In case (1.e) from the construction of $G.p_0$ we have $(G'.T_{\xi_a} \times G'.T_{\xi_b}) \subseteq G'.r_0$, as required.

In case (2) there are five cases to consider: a) $\exists \xi, (a, b) \in G.E_{\xi}$; or b) $a, b \in G.N'_{\xi}$; or c) $a \in G.N'_{\xi} \text{ and } b \in G.E_{\xi}$; or d) $a \in G.E_{\xi} \text{ and } b \in G.N'_{\xi}$; or e) $a \in G.E_{\xi}$, $b \in G.E_{\xi}$, and $\xi_a \neq \xi_b$. Case (2.a) holds vacuously as $(a, b) \in G.r_0 = \emptyset$. In case (2.b) from the construction of $G.r_0$ we have $(a, b) \in G'.r_0$, as required. In case (2.c) from the construction of $G.r_0$ we have $\{\{x\} \times G'.T_{\xi_a}\} \subseteq G'.(\langle N'_{\xi}; r; st\rangle) \subseteq G'.r_{si-hb}$, as required. In case (2.d) from the construction of $G.r_0$ we then have $(\{w\} \times G'.T_{\xi_a}) \subseteq G'.r_{si-hb}$, as required.

In case (3) there are five cases to consider: a) $\exists \xi, (a, b) \in G.E_{\xi}$; or b) $a, b \in G.N'_{\xi}$; or c) $a \in G.N'_{\xi} \text{ and } b \in G.E_{\xi}$; or d) $a \in G.E_{\xi}$ and $b \in G.N'_{\xi}$; or e) $a \in G.E_{\xi}$, $b \in G.E_{\xi}$, and $\xi_a \neq \xi_b$. In case (3.a) from the construction of $G.l_0$ we have $(a, b) \in G.p_0$, as required. Cases (3.b-3.d) hold vacuously as there are no edge to or from non-transactional events. In case (3.e) from the construction of $G.l_0$ we have $(a, b) \in G.p_0$, as required. In case (3.e.i) we have $G'.T_{\xi_a} \times G'.T_{\xi_b} \subseteq G'.(r_{si-hb}) \subseteq \{w\} \times G'.T_{\xi_a}$, as required. In case (3.e.ii) we have $G'.T_{\xi_a} \times G'.T_{\xi_b} \subseteq G'.r_{si-hb}$, as required. In case (3.e.iii) we then have $stg(a, \xi_a) \leq 2$, $stg(b, \xi_b) \geq 3$, and $G'.(T_{\xi_a} \cap R_E) \times G'.(T_{\xi_b} \cap W) \subseteq G'.r_{si-rb}$, as required.
**Inductive case** $n = m+1$

\[
\forall i \in \mathbb{N}, \forall a, b, \xi_a, \xi_b, i \leq m \land (a, b) \in G.\text{hb}_i \Rightarrow \\
(\exists \xi. a, b \in G.E_\xi \Rightarrow (a, b) \in G.\text{po}) \\
\land \left( \neg \exists \xi. a, b \in G.E_\xi \Rightarrow \\
\exists A, B, \emptyset \subset A \times B \subset G', \text{rsi-hb} \\
\land (a \in G.NT \Rightarrow A = \{a\}) \land (b \in G.NT \Rightarrow B = \{b\}) \\
\land [a \in G.E_{\xi_a} \Rightarrow \\
A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.T_{\xi_a} \cap R_E) \\
\lor (\text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W. a \xrightarrow{G,\text{po}} d \land A = \{\text{mw}^{-1}(d)\}) ] \\
\land [b \in G.E_{\xi_b} \Rightarrow \\
B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.T_{\xi_b} \cap W) ] \right) \\
\right)
\]

(I.H.)

Since $(a, b) \in \text{hb}_n$, from the definition of $\text{hb}_n$ we know there exists $c$ such that $(a, c) \in \text{hb}_0$ and $(c, b) \in \text{hb}_n$. There are then five cases to consider: 1) $\exists \xi. (a, b) \in G.E_\xi$; or 2) $a, b \in G.NT$; or 3) $a \in G.NT$ and $b \in G.E_{\xi_a}$; or 4) $a \in G.E_{\xi_a}$ and $b \in G.NT$; or 5) $a \in G.E_{\xi_a}, b \in G.E_{\xi_b}$ and $\xi_a \neq \xi_b$.

**Case 1**

In case (1) pick arbitrary $\xi$ such that $a, b \in G.E_\xi$. There are then three additional cases to consider: a) $c \in G.NT$; b) $c \in G.E_\xi$; or c) there exists $\xi' \neq \xi$ such that $c \in G.E_\xi$.

In case (1.a) from the proof of the base case and the (I.H.) we know there exist $A, B, C \neq \emptyset$ such that $C = \{c\}$, $A \times C \subseteq G'.\text{rsi-hb}$, $C \times B \subseteq G'.\text{rsi-hb}$ and thus $A \times B \subseteq G'.\text{rsi-hb}$ and either: i) $A = G'.T_{\xi}$ and $B = G'.T_{\xi}$; or ii) $A = G'.T_{\xi}$ and $B = (G'.(T_{\xi} \cap W))$; or iii) $\text{stg}(a, \xi) \leq 2$, $A = G'.(T_{\xi} \cap R_E)$ and $B = G'.T_{\xi}$; or iv) $\text{stg}(a, \xi) \leq 3$ and $B = G'.(T_{\xi} \cap W)$; or v) $\text{stg}(a, \xi) \leq 4$, $\exists d \in \xi.W. a \xrightarrow{G,\text{po}} d \land A = \{\text{mw}^{-1}(d)\}$ and $B = G'.T_{\xi}$; or vi) $\text{stg}(a, \xi) \leq 4$, $\exists d \in \xi.W. a \xrightarrow{G,\text{po}} d \land A = \{\text{mw}^{-1}(d)\}$ and $B = G'.(T_{\xi} \cap W)$.

Case (i) cannot arise as we would have $A = B$ and thus $A \times A \subseteq G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent. Case (ii) cannot arise as we would have $(G'.(T_{\xi} \cap W) \times (G'.(T_{\xi} \cap W)) \subseteq G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent. Case (iii) cannot arise as we would have $(G'.(T_{\xi} \cap R_E) \times (G'.(T_{\xi} \cap R_E)) \subseteq G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent. In case (iv) since we have $\text{stg}(a, \xi) \leq 2$ and $\text{stg}(b, \xi) \geq 3$, from the definition of $\text{stg}(\ldots)$ and the construction of $G$ we have $(a, b) \in G.\text{po}$, as required. Cases (v-vi) cannot arise as we would have $\exists d \in \xi.W. (\text{mw}^{-1}(d), \text{mw}^{-1}(d)) \in G'.\text{rsi-hb}$, contradicting the assumption that $G'$ is RSI-consistent.

In case (1.b) from the proof of the base case we have $(a, c) \in G.\text{po}$. On the other hand from (I.H.) we have $(c, b) \in G.\text{po}$. As $G.\text{po}$ is transitively closed, we
have \((a, b) \in G.\text{po}\), as required.

In case (1.c) from the proof of the base case (in cases 1.e, 2.e and 3.e) we have either A) \(G'.T_b \times G'.T_c \subseteq G'.\text{rsi-hb}\); or B) \(\text{stg}(a, \xi) \leq 2, \text{stg}(c, \xi') \geq 3,\) and \(\emptyset \subseteq G'.(T_b \cap \mathcal{R}_E) \times G'.(T_c \cap \mathcal{W}) \subseteq G'.\text{rsi-hb}\). On the other hand from (I.H.) we know there exist \(C, B \neq \emptyset\) such that \(C \times B \subseteq G'.\text{rsi-hb}\) and either: i) \(C = G'.T_b\); or ii) \(C = G'.T_c\), as required. Case (B.i) leads to a contradiction, as otherwise we would have \(G'.\text{rsi-hb}\), as required. Case (B.ii) cannot arise as otherwise we would have \(\emptyset \subseteq G'.(T_c \cap \mathcal{R}_E) \times G'.(T_b \cap \mathcal{W}) \subseteq G'.\text{rsi-hb}\), as required. Case (B.iii) we then have \(\emptyset \subseteq G'.(T_b \cap \mathcal{R}_E) \times \{\text{wu}^{-1}(d)\}\). On the other hand, from (I.H.) we know there exists \(C, B \neq \emptyset\) such that \(C \times B \subseteq G'.\text{rsi-hb}\) and \(B = G'.T_c\); or iv) \(\text{stg}(c, \xi') \leq 2, C = G'.(T_c \cap \mathcal{R}_E)\) and \(B = G'.T_b\); or vi) \(\text{stg}(c, \xi') \leq 4, C = G'.(T_b \cap \mathcal{R}_E)\), \(\text{stg}(b, \xi) \leq 3\) and \(B = G'.(T_c \cap \mathcal{W})\); or vii) \(\text{stg}(c, \xi') \leq 4, C = G'.(T_b \cap \mathcal{R}_E)\), \(\text{stg}(b, \xi) \leq 3\) and \(B = G'.(T_c \cap \mathcal{W})\).

Cases (A.i-A.vi) lead to a cycle in \(G'.\text{rsi-hb}\), contradicting the assumption that \(G'\) is RSI-consistent. In cases (B.ii, B.iv, B.vi) we then have \(\text{stg}(a, \xi) \leq 2, \text{stg}(c, \xi') \geq 3\). Consequently from the definition of \(\text{stg}(\cdot, \cdot)\) and the construction of \(G\) we have \((a, b) \in G.\text{po}\), as required.

Case 2

There are two additional cases to consider: a) \(c \in G.\mathcal{N}T\); or b) there exists \(\xi\) such that \(c \in G.E_\xi\).

In case (2.a) from the proof of the base case we have \((a, c) \in G'.\text{rsi-hb}\). On the other hand from (I.H.) we have \((c, b) \in G'.\text{rsi-hb}\). As \(G'.\text{rsi-hb}\) is transitively closed, we have \((a, b) \in G'.\text{rsi-hb}\), as required.

In case (2.b) from the proof of the base case we know there exists \(C_1 \neq \emptyset\) such that \(\{a\} \times C_1 \subseteq G'.\text{rsi-hb}\) and either: A) \(C_1 = G'.T_b\); or B) \(\text{stg}(c, \xi) \geq 3\) and \(C_1 = G'.(T_c \cap \mathcal{W})\). On the other hand, from (I.H.) we know there exists \(C_2 \neq \emptyset\) such that \(C_2 \times \{b\} \in G'.\text{rsi-hb}\) and either: i) \(C_2 = G'.T_c\); or ii) \(\text{stg}(c, \xi') \leq 2\) and \(C_2 = G'.(T_c \cap \mathcal{R}_E)\); or iii) \(\text{stg}(c, \xi') \leq 4\) and \(\exists d \in \xi, W, c \rightarrow^{G.\text{po}} d \wedge C_2 = \{\text{wu}^{-1}(d)\}\). In cases (A.i-A.iii) and (B.i, B.iii) from the transitivity of \(G'.\text{rsi-hb}\) we have \((a, b) \in G'.\text{rsi-hb}\), as required. Case (B.ii) cannot arise as otherwise we would have \(3 \leq \text{stg}(c, \xi) \leq 2\), leading to a contradiction.

Case 3

There are three additional cases to consider: a) \(c \in G.\mathcal{N}T\); b) \(c \in G.E_{\xi_b}\); or c) there exists \(\xi_c \neq \xi_b\) such that \(c \in G.E_{\xi_c}\).

In case (3.a) from the proof of the base case we have \((a, c) \in G'.\text{rsi-hb}\). On the other hand from (I.H.) we know there exists \(B \neq \emptyset\) such that \(\{c\} \times B \subseteq G'.\text{rsi-hb}\) and \(B = G'.T_{c_b} \cup (\text{stg}(b, \xi_b) \geq 3 \wedge B = G'.(T_{c_b} \cap \mathcal{W}))\). As \(G'.\text{rsi-hb}\) is transitively closed we then know there exists \(B \neq \emptyset\) such that \(\{a\} \times B \subseteq G'.\text{rsi-hb}\) and \(B = G'.T_{c_b} \cup (\text{stg}(b, \xi_b) \geq 3 \wedge B = G'.(T_{c_b} \cap \mathcal{W}))\), as required.
In case (3.b) from the proof of the base case we know there exists \( B \neq \emptyset \) such that \( \{a\} \times B \subseteq G'.\text{rsi-hb} \) and \( B = G'.T_{\xi_a} \lor (\text{stg}(c, \xi_a) \geq 3 \land B = G'.(T_{\xi_a} \cap W)) \). On the other hand, from (I.H.) we have \( c \xrightarrow{G.\text{po}} b \) and thus from the definition of \( \text{stg}(\cdot, \cdot) \) and the construction of \( G \) we have \( \text{stg}(b, \xi_b) \geq \text{stg}(c, \xi_c) \). As such, we know there exists \( B \neq \emptyset \) such that \( \{a\} \times B \subseteq G'.\text{rsi-hb} \) and \( B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W)) \), as required.

In case (3.c) from the proof of the base case we know there exists \( C_1 \neq \emptyset \) such \( \{a\} \times C_1 \subseteq G'.\text{rsi-hb} \) and either \( A \) \( C_1 = G'.T_{\xi_a} \); or \( B \) \( \text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_a} \cap W) \). On the other hand, from (I.H.) we know there exist \( C_2 \neq \emptyset \) such that \( C_2 \times B \in G'.\text{rsi-hb} \) and either: i) \( C_2 = G'.T_{\xi_c} \) and \( B = G'.T_{\xi_b} \); or ii) \( C_2 = G'.T_{\xi_c} \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_a} \cap W) \); or iii) \( \text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_c} \cap R_E) \) and \( B = G'.T_{\xi_b} \); or iv) \( \text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_c} \cap R_E) \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W) \); or v) \( \text{stg}(c, \xi_c) \leq 4 \land \exists d \in \xi_c.W \). \( c \xrightarrow{G.\text{po}} d \wedge C_2 = \{\text{mw}^{-1}(d)\} \) and \( B = G'.T_{\xi_b} \); or vi) \( \text{stg}(c, \xi_c) \leq 4 \land \exists d \in \xi_c.W \). \( c \xrightarrow{G.\text{po}} d \wedge C_2 = \{\text{mw}^{-1}(d)\} \) and \( \text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W) \).

In cases (A.i-A.iv) and (B.i, B.ii, B.v, B.vi) from the transitivity of \( G'.\text{rsi-hb} \) we know there exists \( B \neq \emptyset \) such that \( \{a\} \times B \subseteq G'.\text{rsi-hb} \) and \( B = G'.T_{\xi_b} \lor (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W)) \), as required. Cases (B.iii, B.iv) cannot arise as we would otherwise have \( 3 \leq \text{stg}(c, \xi_c) \leq 2 \), leading to a contradiction.

Case 4
There are three additional cases to consider: a) \( c \in G.N.T \); b) \( c \in G.E_{\xi_a} \); or c) there exists \( \xi \neq \xi_a \) such that \( c \in G.E_{\xi} \).

In case (4.a) from (I.H.) we have \( (c, b) \in G'.\text{rsi-hb} \). On the other hand from the base case we know there exists \( A \neq \emptyset \) such that \( A \times \{c\} \subseteq G'.\text{rsi-hb} \) and \( A = G'.T_{\xi_c} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \lor \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W \). \( a \xrightarrow{G.\text{po}} d \wedge A = \{\text{mw}^{-1}(d)\} \). As \( G'.\text{rsi-hb} \) is transitively closed we then know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.\text{rsi-hb} \) and \( A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \lor \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W \). \( a \xrightarrow{G.\text{po}} d \wedge A = \{\text{mw}^{-1}(d)\} \), as required.

In case (4.b) from (I.H.) we know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.\text{rsi-hb} \) and \( A = G'.T_{\xi_b} \lor (\text{stg}(c, \xi_c) \leq 2 \land A = G'.(T_{\xi_c} \cap R_E) \lor \text{stg}(c, \xi_c) \leq 4 \land \exists d \in \xi_c.W \). \( a \xrightarrow{G.\text{po}} d \wedge A = \{\text{mw}^{-1}(d)\} \). On the other hand, from the proof of the base case we have \( a \xrightarrow{G.\text{po}} c \) and thus from the definition of \( \text{stg}(\cdot, \cdot) \) and the construction of \( G \) we have \( \text{stg}(a, \xi_a) \leq \text{stg}(c, \xi_c) \). As such, we know there exists \( A \neq \emptyset \) such that \( A \times \{b\} \subseteq G'.\text{rsi-hb} \) and \( A = G'.T_{\xi_a} \lor (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \lor \text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W \). \( a \xrightarrow{G.\text{po}} d \wedge A = \{\text{mw}^{-1}(d)\} \), as required.

In case (4.c) from the proof of the base case we know there exist \( A, C_1 \neq \emptyset \) such that \( A \times C_1 \subseteq G'.\text{rsi-hb} \) and either: i) \( A = G'.T_{\xi_a} \) and \( C_1 = G'.T_{\xi_c} \); or ii) \( A = G'.T_{\xi_c} \) and \( (\text{stg}(c, \xi_c) \geq 3 \land C_1 = G'.(T_{\xi_a} \cap W)) \); or iii) \( \text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \) and \( C_1 = G'.T_{\xi_c} \); or iv) \( \text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E) \) and
(stg(c, ξ_c) ≥ 3 ∧ C_1 = G'.(\mathcal{T}_{ξ_c} \cap W)); or v) stg(a, ξ_a) ≤ 4 ∧ 3d ∈ ξ_a.W. a \xrightarrow{G'.po} d ∧ A = \{\text{rsi-hb}\} and either A) C_2 = G'.\mathcal{T}_{ξ_c}; or B) (stg(c, ξ_c) ≤ 2 ∧ C_2 = G'.(\mathcal{T}_{ξ_c} \cap W)); or C) (stg(a, ξ_a) ≤ 4 ∧ 3d ∈ ξ_a.W. c \xrightarrow{G'.po} d ∧ A = \{\text{rsi-hb}\}).

On the other hand, from (I.H.) we know there exists C_2 \neq \emptyset such that C_2 \times \{b\} \subseteq G'.rsi-hb and either A) C_2 = G'.\mathcal{T}_{ξ_c}; or B) (stg(c, ξ_c) ≤ 2 ∧ C_2 = G'.(\mathcal{T}_{ξ_c} \cap W)); or C) (stg(a, ξ_a) ≤ 4 ∧ 3d ∈ ξ_a.W. c \xrightarrow{G'.po} d ∧ A = \{\text{rsi-hb}\}).

In cases (A.i-A.vi), (C.i-C.vi) and (B.i, B.iii, B.v) from the transitivity of G'.rsi-hb we know there exists A \neq \emptyset such that A \times \{b\} \subseteq G'.rsi-hb and A = G'.\mathcal{T}_{ξ_c} ∨ (stg(a, ξ_a) ≤ 2 ∧ A = G'.(\mathcal{T}_{ξ_c} \cap W)).

Case 5

There are four additional cases to consider: a) c ∈ G.N'T; b) c ∈ G.E_{ξ_a}; or c) c ∈ G.E_{ξ_c}; or d) there exists ξ_c such that ξ_c \neq ξ_a, ξ_c \neq ξ_b and c ∈ G.E_{ξ_c}.

In case (5.a) from the proof of the base case we know there exists A \neq \emptyset such that A \times \{c\} \subseteq G'.rsi-hb and A = G'.\mathcal{T}_{ξ_c} ∨ (stg(a, ξ_a) ≤ 2 ∧ A = G'.(\mathcal{T}_{ξ_c} \cap W)). On the other hand from (I.H.) we know there exists B \neq \emptyset such that \{c\} \times B \subseteq G'.rsi-hb and B = G'.\mathcal{T}_{ξ_c} ∨ (stg(b, ξ_b) ≥ 3 ∧ B = G'.(\mathcal{T}_{ξ_c} \cap W)). As such, since G'.rsi-hb is transitive we know there exist A, B \neq \emptyset such that A \times B \subseteq G'.rsi-hb; that A = G'.\mathcal{T}_{ξ_c} ∨ (stg(a, ξ_a) ≤ 2 ∧ A = G'.(\mathcal{T}_{ξ_c} \cap W)) ∨ (stg(b, ξ_b) ≥ 3 ∧ B = G'.(\mathcal{T}_{ξ_c} \cap W)), as required.

In case (5.b) from (I.H.) we know there exist A, B \neq \emptyset such that A \times B \subseteq G'.rsi-hb; that A = G'.\mathcal{T}_{ξ_c} ∨ (stg(a, ξ_a) ≤ 2 ∧ A = G'.(\mathcal{T}_{ξ_c} \cap W)) ∨ (stg(c, ξ_c) ≤ 4 ∧ 3d ∈ ξ_a.W. a \xrightarrow{G'.po} d ∧ A = \{\text{rsi-hb}\}); and that B = G'.\mathcal{T}_{ξ_c} ∨ (stg(b, ξ_b) ≥ 3 ∧ B = G'.(\mathcal{T}_{ξ_c} \cap W)), as required.

In case (5.c) from the proof of the base case we know there exist A, B \neq \emptyset such that A \times B \subseteq G'.rsi-hb; that A = G'.\mathcal{T}_{ξ_c} ∨ (stg(a, ξ_a) ≤ 2 ∧ A = G'.(\mathcal{T}_{ξ_c} \cap W)) ∨ (stg(c, ξ_c) ≤ 4 ∧ 3d ∈ ξ_a.W. a \xrightarrow{G'.po} d ∧ A = \{\text{rsi-hb}\}); and that B = G'.\mathcal{T}_{ξ_c} ∨ (stg(b, ξ_b) ≥ 3 ∧ B = G'.(\mathcal{T}_{ξ_c} \cap W)), as required.
We proceed by contradiction. Let us assume that there exist $A,C_1 \neq \emptyset$ such that $A \times C_1 \subseteq G'.rsi-hb$; that $A = G'.T_{\xi_a} \cup (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E))$, as required.

In case (5.3) from the proof of the base case we know there exist $A_C \neq \emptyset$; that $A \times C_1 \subseteq G'.rsi-hb$; that $A = G'.T_{\xi_a} \cup (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)), as required.

On the other hand, from (I.H.) we know there exist $C_2, B \neq \emptyset$ such that $C_2 \times B \subseteq G'.rsi-hb$; that $B = G'.T_{\xi_b} \cup (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))$; and that either: i) $C_2 = G'.T_{\xi_c}$; or ii) $\text{stg}(c, \xi_c) \leq 2 \land C_2 = G'.(T_{\xi_c} \cap R_E)$; iii) $\text{stg}(c, \xi_c) \leq 4 \land \exists e \in \xi_c.W, c \overset{G'.po}{\rightarrow} e \land C_2 = \{\text{mw}^{-1}(e)\}$.

In cases (A.i-A.iii) and (B.i, B.iii) from the transitivity of $G'.rsi-hb$ we know there exists $A, B \neq \emptyset$ such that $A \times B \subseteq G'.rsi-hb$; that $A = G'.T_{\xi_a} \cup (\text{stg}(a, \xi_a) \leq 2 \land A = G'.(T_{\xi_a} \cap R_E)) \lor (\text{stg}(a, \xi_a) \leq 4 \land \exists d \in \xi_a.W, a \overset{G'.po}{\rightarrow} d \land A = \{\text{mw}^{-1}(d)\})$; and that $B = G'.T_{\xi_b} \cup (\text{stg}(b, \xi_b) \geq 3 \land B = G'.(T_{\xi_b} \cap W))$, as required. Case (B.ii) cannot arise as we would otherwise have $3 \leq \text{stg}(c, \xi_c) \leq 2$, leading to a contradiction.

**Theorem 14 (Completeness).** For all RSI execution graphs $G'$ and their counterpart implementation graphs $G$ constructed as above,

$$\text{rsi-consistent}(G') \Rightarrow \text{RA-consistent}(G)$$

**Proof.** Pick an arbitrary RSI execution graph $G'$ and its counterpart implementation graph $G$ constructed as above and let us assume that $\text{rsi-consistent}(G')$ holds.

From the definition of $\text{RA-consistent}(G)$ it then suffices to show:

1. irreflexive($Ghb$)
2. irreflexive($Gmo, Ghb$)
3. irreflexive($Gtb, Ghb$)

**RTS. part 1**

We proceed by contradiction. Let us assume that there exists $a$ such that $(a, a) \in Ghb$. There are now two cases to consider: 1) $a \in G.T$; or 2) $\exists \xi, a \in G.E_\xi$

In case (1) from Lemma 19 we have $(a, a) \in G'.rsi-hb$, contradicting the assumption that $G'$ is RSI-consistent. Similarly, in case (2) from Lemma 19 we have $(a, a) \in G.po$, leading to a contradiction as $G.po$ is acyclic by construction.

**RTS. part 2**

We proceed by contradiction. Let us assume that there exist $a, b$ such that $(a, b) \in Ghb$ and $(b, a) \in Gmo$. Let $\text{loc}(a) = \text{loc}(b) = x$. There are then five cases to consider: 1) $\exists \xi, (a, b) \in G.E_\xi$; or 2) $a, b \in G.T$; or 3) $a \in G.T$ and $b \in G.E_\xi$; or 4) $a \in G.E_\xi$ and $b \in G.T$; or 5) $a \in G.E_\xi$, $b \in G.E_\xi$ and $\xi_a \neq \xi_b$.

In case (1) from the construction of $Gmo$ we have $(\text{mw}^{-1}(b), \text{mw}^{-1}(a)) \in G'.mo$. Moreover, from Lemma 19 we have $(a, b) \in G.po$, and thus from the
We proceed by contradiction. Let us assume that there exists \( mw_1(a), mw_1(b) \) such that \( G^\prime .po \) and \( G^\prime .mo \) contradict the assumption that \( G^\prime \) is RSI-consistent.

In case (2) from the construction of \( G\).po we have \( (a,b) \in G^\prime .po \). Moreover, since \( (a,b) \) are write events in the same transaction \( \xi \), we have \( \exists a, b \in G^\prime .po \cap W^2 \subseteq G^\prime .rsi-hb \). As such we have \( \forall a \in G^\prime .po \) \( b \in G^\prime .mo \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

In case (3) from the construction of \( G\).po we have \( (a,b) \in G^\prime .mo \). Moreover, since \( (a,b) \) are write events in the same transaction \( \xi \), we have \( \exists a, b \in G^\prime .po \cap W^2 \subseteq G^\prime .rsi-hb \). As such we have \( \forall a \in G^\prime .po \) \( b \in G^\prime .mo \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

In cases (4, 5) from the construction of \( G \) we know that \( mw_1(a) \xrightarrow{G,po} b \) and thus from the transitivity of \( G \) we have \( (a,b) \in G^\prime .hb \). As such we have \( \forall a \in G^\prime .po \) \( b \in G^\prime .mo \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

RTS, part 3

We assume that there exists \( a, b \) such that \( (a,b) \in G.hb \) and \( (a,b) \in G.rb \). Let \( \text{loc}(a) = \text{loc}(b) = \alpha \). There are then five cases to consider: 1) \( \exists \xi, (a,b) \in G.E_\xi \); or 2) \( a, b \in G.NT \); or 3) \( a \in G.NT \) and \( b \in G.E_\xi \); or 4) \( a \in G.E_\xi \) and \( b \in G.NT \); or 5) \( a \in G.E_\xi \) and \( b \in G.E_\xi \) and \( \xi_a \neq \xi_b \).

Case (1) cannot arise as from the definition of \( G.rb \) we know \( a \) is a write event in \( \xi.Ws \) while \( b \) is a read event in \( \xi.Ts \) and no po edge exists between the events of \( \xi.Ts \) and \( \xi.Ws \).

In case (1) from Lemma 19 we have \( (a,b) \in G.po \). This however leads to a contradiction as from the definition of \( G.rb \) we know \( a \) is a write event in \( \xi.Ws \) while \( b \) is a read event in \( \xi.Ts \) and no po edge exists between the events of \( \xi.Ts \) and \( \xi.Ws \).

In case (2) from the construction of \( G.rb \) we know that \( (a,b) \in G^\prime .rb \). On the other hand, from Lemma 19 we have \( (a,b) \in G^\prime .rsi-hb \). As such we have \( a \xrightarrow{G^\prime .rsi-hb} b \xrightarrow{G^\prime .rb} a \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

In case (3), as \( b \) is a read event and thus \( \text{stg}(b, \xi_b) = 1 \), from Lemma 19 we have \( (a,b) \in G^\prime .rsi-hb \). On the other hand, from the construction of \( G.rb \) we have \( (a,b) \in G^\prime .rb \). Consequently, we have \( a \xrightarrow{G^\prime .rsi-hb} b \xrightarrow{G^\prime .rb} a \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

In case (4) since \( a \) is a write, from the construction of \( G \) we know that \( mw_1(a) \xrightarrow{G,po} b \) and thus from the transitivity of \( G.hb \) we have \( (a,b) \in G.hb \). As such we have \( \forall a \in G^\prime .po \) \( b \in G^\prime .mo \), contradicting the assumption that \( G^\prime \) is RSI-consistent.

In case (5) since \( a \) is a write, from the construction of \( G \) we know that \( mw_1(a) \xrightarrow{G,po} b \) and thus from the transitivity of \( G.hb \) we have \( (a,b) \in G.hb \). As such we have \( \forall a \in G^\prime .po \) \( b \in G^\prime .mo \), contradicting the assumption that \( G^\prime \) is RSI-consistent.
Consequently, we have \( mw^{-1}(a) \xrightarrow{G', rsi-hb} b \xrightarrow{G', rb} mw^{-1}(a) \), contradicting the assumption that \( G' \) is RSI-consistent.

Similarly, in case (5) since \( a \) is a write event, from the construction of \( G \) we know that \( mw^{-1}(a) \xrightarrow{G, po} a \xrightarrow{G, hb} b \), and thus from the transitivity of \( G, hb \) we have \((mw^{-1}(a), b) \in G, hb \). As \( stg(mw^{-1}(a), \xi_a) = stg(b, \xi_b) = 1 \), from Lemma 19 we have \((mw^{-1}(a), b) \in G', rsi-hb \). On the other hand, from the construction of \( G, rb \) we have \((b, mw^{-1}(a)) \in G', rb \). Consequently, we have \( mw^{-1}(a) \xrightarrow{G', rsi-hb} b \xrightarrow{G', rb} mw^{-1}(a) \), contradicting the assumption that \( G' \) is RSI-consistent.