Christian Tilgner
Boris Glavic
Michael Boehlen
Carl-Christian Kanne

Correctness proof of the declarative SS2PL protocol implementation

TECHNICAL REPORT – No. 2010.IFI-2010.0008

September 2010

C. Tilgner
B. Glavic
M. Boehlen
C. Kanne:
Technical Report No. 2010.IFI-2010.0008, September 2010
Database Technology
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.ifi.uzh.ch/groups/dbtg/index.html

Correctness Proof of the Declarative SS2PL Protocol Implementation

Christian Tilgner[◇] Boris Glavic[♠] Michael Böhlen[◇] Carl-Christian Kanne[♣]

University of Zurich, Switzerland[◇]
{tilgner,boehlen}@ifi.uzh.ch

University of Toronto, Canada[♠]
glavic@cs.toronto.edu

University of Mannheim, Germany[♣]
kanne@informatik.uni-mannheim.de

1. INTRODUCTION

In [1], we proposed a novel approach for scheduling. The underlying idea of our declarative scheduling approach is to **(1)** treat sets of requests as data collections and to **(2)** employ database query processing techniques over this request data to produce high-quality schedules in an efficient and flexible manner. Scheduling protocols are implemented as queries that select those requests from the request data collections whose execution does not violate the scheduling constraints (e.g. correctness criteria and service-level agreements).

We denote such queries implementing scheduling protocols declaratively as *declarative protocol implementations (DPI)*. DPIs are much more concise, easier to understand and easier to modify than an imperative scheduler implementation. Our approach allows for an easy definition of scheduling protocols of different categories such as traditional lock-based, application-specific consistency, QoS-based and multiversioning scheduling protocols. Additionally, declaratively formulated scheduling constraints are specified very close to their formal definition and allow to prove the correctness of such declarative scheduling protocol implementations.

In this technical report, we illustrate the way of proving DPIs on the basis of the strong two-phase locking (SS2PL) protocol (also known as rigorous 2PL). For the DPIs of other protocols, the proof can be done in an equivalent manner.

The remainder of this technical report is as follows: We present the core ideas of our declarative scheduling approach in Section 1.1 and give the definition of SS2PL in Section 1.2. Section 1.3 covers necessary notational remarks. The SS2PL DPI and its correctness proof are given in Section 1.4 resp. Chapter 2.

1.1 Declarative Scheduling

In this work, we limit the discussion to scheduling atomic database requests such as read operations of a transaction i on object A denoted as $r_i(A)$, write operations $w_i(A)$, abort operations a_i and commit operations c_i. To store these requests in a database, we make use of the database schema illustrated in Figure 1. An attribute description is given in Table 1.

$$\begin{array}{ll}\mathcal{R} & \text{(TA, Seq, Op, Ob)}\\ \mathcal{H} & \text{(ID, TA, Seq, Op, Ob)}\\ \mathcal{E} & \text{(ID, TA, Seq, Op, Ob)}\end{array}$$

Figure 1: Schema of Declarative Scheduler

To ease the presentation, we proof the correctness of the declarative SS2PL protocol implementation on the basis of this basic database schema. We omit other possible non-key attributes which are not constrained.

Table 1: Attribute Descriptions

Attribute	Description
ID	Consecutive request ID
TA	Transaction ID
Seq	Request sequence within a transaction
Op	Operation type (read/write/abort/commit)
Ob	Object ID

The declarative scheduler stores all new requests sent by the clients in relation $\mathcal{R}$ which acts as a buffer for requests that have to be scheduled for execution. Relation $\mathcal{H}$ comprises prior executed requests in their execution order, i.e., it stores the schedule that has been generated so far. It is needed because most scheduling constraints cannot be evaluated without information about the generated schedule. *Example:* Classical implementations of lock-based protocols store the locks hold by transactions explicitly. In the declarative implementation of such protocols we do not store those locks explicitly. Instead, we use query processing to determine on the fly which transactions *logically* hold locks on which objects based solely on information in relation $\mathcal{H}$. Relation $\mathcal{E}$ is used to buffer requests that have already been chosen for execution.

The declarative scheduler generates a schedule for incoming client requests by applying the algorithm presented in Figure 2. The algorithm consists of seven steps which are repeated infinitely. We refer to one execution of the while loop as a *scheduler iteration.* The steps executed during each scheduler iteration are explained in the following: **(1)** All requests which have been executed during the last iteration are deleted from relation $\mathcal{R}$ (line 4). **(2)** All requests send by the clients since the last iteration are inserted into relation $\mathcal{R}$ by $GetNew()$ (line 5). **(3)** Afterwards, requests that cannot be executed during the current and any future scheduler iterations are removed from $\mathcal{R}$ (line 6). The DPI *Revoked* query is executed over relations $\mathcal{R}$ and $\mathcal{H}$ to identify these requests ($Revoked(\mathcal{H}, \mathcal{R})$). **(4)** We execute the *Schedule* query to identity which of the pending requests from relation $\mathcal{R}$ should be executed next. The result of *Schedule* is stored in relation $\mathcal{E}$ (line 7). This means, instead of scheduling request per request we schedule sets of requests (all requests of relation $\mathcal{R}$) at once. **(5)** Once the statements have been inserted into relation $\mathcal{E}$, they get executed statement per statement (line 8). **(6)** These executed requests are inserted into relation $\mathcal{H}$ (line 9). **(7)** All requests that are irrelevant for scheduling decisions (identified by running the DPI query $Irrelevant(\mathcal{H})$) are deleted from relation $\mathcal{H}$ (line 10).

```
1  $\mathcal{H} = \mathcal{E} = \mathcal{R} = \emptyset$
2  while true do
3  begin
4      $\mathcal{R} = \mathcal{R} - \mathcal{E}$;
5      $\mathcal{R} = \mathcal{R} \cup GetNew()$;
6      $\mathcal{R} = \mathcal{R} - Revoked(\mathcal{H}, \mathcal{R})$;
7      $\mathcal{E} = Schedule(\mathcal{H}, \mathcal{R})$;
8      $Execute(\mathcal{E})$;
9      $\mathcal{H} = \mathcal{H} \cup \mathcal{E}$;
10     $\mathcal{H} = \mathcal{H} - Irrelevant(\mathcal{H})$;
11 end
```

Figure 2: Pseudocode of Declarative Scheduler

1.2 Strong 2PL Protocol

As already mentioned, our declarative scheduler is capable of applying various scheduling protocols of different categories. To illustrate the way of proving their correctness, we chose the lock-based scheduling protocol SS2PL. The SS2PL protocol specifies that for each transaction the following constraints 1-5 of the 2PL protocol must hold [2]:

1. An object has to be locked before it can be read or written.
2. A transaction may not acquire a lock if already holding the lock.
3. A transaction has to respect the locks on the relevant object held by other transactions based on the following lock compatibility table [2]:

		Lock requested	
		$s_i(A)$	$x_i(A)$
Lock	$s_j(A)$	+	-
currently	$x_j(A)$	-	-
held	no lock	+	+

   The table is to be read as follows: a requested lock (s denotes shared lock, x denotes exclusive lock) of transaction i on an object A may be granted if it does not conflict (+) with a currently held lock on object A of transaction j ($i \neq j$). Otherwise it conflicts (-) and the requested lock may not be granted.

4. All locking operations of a transaction (including read and write lock operations) have to precede the first unlock operation of this transaction. This defines two phases for each transaction, a growing and a shrinking phase.
5. A transaction has to release all its locks at its end.

The SS2PL protocol additionally specifies a sixth constraint [2]:

6. All locks of a transaction t, including its exclusive write and shared read locks, are held until t terminates, i.e., t releases all its locks not before it commits resp. aborts.

The SS2PL protocol guarantees that every schedule produced by this protocol fulfills these six constraints. Such schedules are always serializable and fulfill the strictness criterion [2].

1.3 Notational Remarks

Our prototype requires DPI queries to be expressed in SQL. However, for reasons of space and readability, all DPI queries are given as domain relational calculus (DRC) expressions in this paper. We use the following notational conventions for DRC: Capital letters denote variables and small letters indicate constants. All variables that are not used in an universal quantification are implicitly assumed to be existentially quantified. For instance, instead of $\exists A, B(I(A, B) \wedge \neg\exists C, D(J(C, D)))$ we write $I(A, B) \wedge \neg J(C, D)$. Unrestricted existentially quantified variables are represented by _ and disjunctive use of constants is represented by |. Thus, for the DRC expression $I(A, B) \wedge (A = a \vee A = c)$ we use the shortcut $I(a|c, \_)$. We define aggregation as:

$$\{G_1, G_2, \ldots, G_n, F_1(A_1), F_2(A_2), \ldots, F_n(A_n) \mid E\}$$

where E is a DRC expression, $G_1, \ldots, G_n$ are attributes on which to group (can be empty), and each F_i is an aggregate function over attribute A_i. We make use of set functions such as union $\cup$ and difference $-$.

The use of DRC instead of SQL is unproblematic, because the translation of a DRC expression into an SQL statement is straightforward. For example, the SQL query presented below selects all operations of already committed transactions:

```
SELECT TA, Seq ,Op,Ob
FROM H as h1
WHERE EXISTS (   SELECT *
                 FROM H as h2
                 WHERE h1.TA=h2.TA AND
                 (h2.Op='a' OR h2.Op='c')
)
```

This query can easily and shorter be formulated in DRC as:

$$\{T, N, A, O | \mathcal{H}(\_, T, N, A, O, \_, \_) \wedge \mathcal{H}(\_, T, \_, a|c, \_, \_, \_)\}$$

1.4 Declarative SS2PL Protocol Implementation

The tasks of scheduling requests and updating history information is modeled by iteratively applying the following three queries, called the *declarative protocol implementation* or *DPI*, that implement these tasks for the scheduling protocol of the application:

- **Schedule**: Identifies which requests should be executed next without breaking any of the scheduling constraints (e.g. strong consistency) based on historic request data.
- **Revoked**: Removed requests of transactions from $\mathcal{R}$ which cannot finish their executions.
- **Irrelevant**: Removes parts of the history that are not relevant anymore for future scheduling decisions.

For the SS2PL DPI, we make use of relations $\mathcal{X}_i$ and $\mathcal{S}_i$. Thereby, i denotes the state of these relations at the end of scheduler run i. $\mathcal{X}_i$ resp. $\mathcal{S}_i$ contain all objects which are write-locked resp. read-locked including the transaction holding the lock. We never materialize these relations. Instead, they are realized as queries. This means locking information is never actually stored, but extracted from relation $\mathcal{H}_i$ on the fly. The DRC formulations of $\mathcal{X}_i$ and $\mathcal{S}_i$ are as follows:

$$\mathcal{X}_{i+1} = \{O, T | \mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)\}$$
$$\mathcal{S}_{i+1} = \{O, T | \mathcal{H}_i(\_, T, \_, r, O) \wedge \neg\mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)\}$$

The SS2PL *Schedule* query is illustrated in Figure 3. The SS2PL *Schedule* query selects those statements from relation $\mathcal{R}$ which can be executed safely without violating the SS2PL constraints defined in Section 1.2. This includes all statements contained in relation $\mathcal{R}$ without two sets of statements: No operations accessing write-locked objects (*OpsOnXLO*) and no write operations accessing read-locked objects (*WOpsOnSLO*) may be selected from relation $\mathcal{R}$. Additionally, only one statement per object (*MinStmtPerObj*) may be selected from relation $\mathcal{R}$. Otherwise, the third SS2PL constraint, defined in Section 1.2, does not hold for the resulting instance of relation $\mathcal{H}$. Aborts as well as commits may always be selected for execution.

MinStmtPerObj is selecting only one statement per operation because of the following reason: Assume $\mathcal{R}$ contains two operations $w_1(A)$ and $w_2(A)$ that want to write an unlocked object A. If both requests get chosen by the SS2PL *Schedule* query, this would result in a conflict with constraint 3 in Section 1.2. Even if the selection of the two operations $r_3(A)$ and $r_4(A)$ does not lead to a violation, for simplicity we decided for the more strict strategy to choose only one statement per object (e.g. $r_3(A)$). The other operations (e.g. $r_4(A)$) will be considered at the next scheduler run. But the *Schedule* query can easily be extended to be able to handle multiple read operations on the same object in the same scheduler run.

The statements selected from *Schedule* get inserted into H: $\mathcal{H} = \mathcal{H} \cup \mathcal{E}$. Thereby, a unique ID is assigned to every statement (GenID()) to establish a total request order in $\mathcal{H}$.

2. CORRECTNESS PROOF

In this subsection, we prove that the DRC formulation of the SS2PL protocol, shown in Figure 3, fulfills the six correctness conditions outlined in Section 1.2. To prove this claim, we have to show that the state of relation $\mathcal{H}$ produced by each scheduler iteration conforms to conditions 1-6 of the SS2PL protocol presented in Section 1.2.

At first, we recursively define the state of the request database relations (RDB) after scheduler iteration $i+1$ based on scheduler iteration i and the initial state of these relations to show the development of relation $\mathcal{H}_i$. Afterwards, we translate conditions 1-6 presented in Section 1.2 into a DRC formulation C1-C6 over relation $\mathcal{H}$ and define what it means that a state of relation $\mathcal{H}$ respects the SS2PL protocol.

Definition 1 (Request database state). *For an input sequence* $<New_0, \ldots, New_i>$*,* $i \in N$*, we call the tuple* $RDB_i = (\mathcal{H}_i, \mathcal{R}_i, \mathcal{E}_i, New_i)$ *the request database state after scheduler iteration* i*.* RDB_i *is defined recursively:*

$$i \leq 0 : \mathcal{H}_i = \mathcal{E}_i = \mathcal{R}_i = New_i = \emptyset$$

$$i > 0 : \mathcal{R}_{i+1} = \mathcal{R}_i \cup New_{i+1} - \{T, N, A, O \mid \mathcal{E}_i(\_, T, N, A, O)\}$$
$$\mathcal{E}_{i+1} = Schedule(\mathcal{R}_{i+1}, \mathcal{H}_i)$$
$$\mathcal{H}_{i+1} = \mathcal{H}_i \cup \mathcal{E}_{i+1}$$

An example illustrating the request database state of six scheduler iterations is given in Figure 4. As defined in Definition 1, the presented relations show the instances *at the end* of the scheduler iteration. For illustration purposes, we did not delete the irrelevant requests from $\mathcal{H}$. Instead, we solely mark them with grey background.

In the DRC translation we often refer to operations that were executed at scheduler iteration i. We use $\mathcal{H}_i^E$ as a shortcut for $\mathcal{H}_i^E = \mathcal{H}_i - \mathcal{H}_{i-1}$ resp. $\{I, T, N, A, O \mid \mathcal{H}_i(I, T, N, A, O) \wedge \neg\mathcal{H}_{i-1}(I, T, N, A, O)\}$ with $i > 0$, i.e., the operations that got included into the history exactly at scheduler run i. We define $\mathcal{H}_i^E = \emptyset$ for $i \leq 0$.

We assume that all transactions respect the following preconditions: **(P1)** If $\mathcal{R}$ contains a request r from transaction T then no further requests of this transaction will be added until request r is processed, i.e., a transaction waits until its current request is executed before issuing new requests. **(P2)** No transaction is taking any actions after its commit or abort.

These preconditions can be expressed in DRC as follows:

$$(\mathbf{P1}) : \forall T, N, i : \mathcal{R}_i(T, N, \_, \_) \Rightarrow \neg\mathcal{R}_i(T, N_2, \_, \_) \wedge N \neq N_2$$
$$(\mathbf{P2}) : \forall T, h, i : h < i \wedge \mathcal{H}_h^E(\_, T, \_, a|c, \_) \Rightarrow \neg\mathcal{H}_i^E(\_, T, \_, \_, \_)$$

Note that P2 is equivalent to the following formulation:

$$\forall h, i, T : h < i \wedge \mathcal{H}_i^E(\_, T, \_, \_, \_) \Rightarrow \neg\mathcal{H}_h^E(\_, T, \_, a|c, \_)$$

Mapping the SS2PL constraints to our declarative approach results in the DRC expressions $C1$-$C6$ presented in the definition below.

Definition 2 (SS2PL protocol). *A sequence of states of relation* $\mathcal{H}$ *respects the SS2PL protocol iff the following conditions hold:*

Condition C1: An object has to be locked before it can be read or written. This condition can be expressed in DRC as:

$$\forall T, O, i : \mathcal{H}_i^E(\_, T, \_, r, O) \Rightarrow \mathcal{S}_{i+1}(O, T) \vee \mathcal{X}_{i+1}(O, T)$$
$$\wedge \forall T, O, i : \mathcal{H}_i^E(\_, T, \_, w, O) \Rightarrow \mathcal{X}_{i+1}(O, T)$$

We do not lock objects in advance. Instead, Schedule does not select requests accessing locked objects. And there can not arise a locking problem between requests selected at the same scheduler run due to MinStmtPerObj as explained in Section 1.4.

Condition C3: A transaction has to respect the locks on the relevant object held by other transactions based on the lock compatibility table.

$$\forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2)$$
$$\wedge \forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{i+1}(O, T_2)$$
$$\wedge \forall O, T, T_2, i : \mathcal{S}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2)$$

Condition C4: All locking operations of a transaction (including read and write lock operations) have to precede the

$$\begin{array}{rcl}
\mathcal{X} & = & \{O,T|\mathcal{H}(\_,T,\_,w,O) \wedge \neg\mathcal{H}(\_,T,\_,a|c,\_)\} \\
\mathcal{S} & = & \{O,T|\mathcal{H}(\_,T,\_,r,O) \wedge \neg\mathcal{H}(\_,T,\_,w,O) \wedge \neg\mathcal{H}(\_,T,\_,a|c,\_)\} \\
\mathit{OpsOnXLO} & = & \{T,N,O|\mathcal{R}(T,N,\_,O) \wedge \mathcal{X}(O,T_2) \wedge T \neq T_2\} \\
\mathit{WOpsOnSLO} & = & \{T,N,O|\mathcal{R}(T,N,w,O) \wedge \mathcal{S}(O,T_2) \wedge T \neq T_2\} \\
\mathit{LegalOps} & = & \{T,N,O|\mathcal{R}(T,N,\_,O)\wedge \\
& & \neg \mathit{OpsOnXLO}(T,N,O) \wedge \neg \mathit{WOpsOnSLO}(T,N,O)\} \\
\mathit{MinStmtPerObj} & = & \{O,\mathit{Min}(T)|\mathit{LegalOps}(T,\_,O)\} \\
\mathit{Schedule} & = & \{\mathit{GenID}(),T,N,A,O|\mathcal{R}(T,N,A,O)\wedge \\
& & (\mathit{MinStmtPerObj}(\_,T) \vee (A=a \vee A=c))\}
\end{array}$$

Figure 3: SS2PL DPI

first unlock operation of this transaction. This defines two phases for each transaction, a growing and a shrinking phase.

This condition is automatically fulfilled by condition C6.

Condition C5: *A transaction has to release all its locks at its end.*

$$\begin{aligned}&\forall T,h,i : h < i+1 \wedge \mathcal{H}_h(\_,T,\_,a|c,\_)\\ &\Rightarrow \neg\mathcal{X}_{i+1}(\_,T) \wedge \neg\mathcal{S}_{i+1}(\_,T)\end{aligned}$$

Condition C6: *All locks of a transaction t, including its exclusive write and shared read locks, are held until t terminates, i.e., t releases all its locks not before it commits resp. aborts.*

$$\begin{aligned}&\forall T,O,g,h,i : g \leq h \leq i+1 \wedge \mathcal{X}_g(O,T) \wedge \neg\mathcal{H}_i(\_,T,\_,a|c)\\ &\Rightarrow \mathcal{X}_h(O,T)\\ &\wedge\forall T,O,g,h,i : g \leq h \leq i+1 \wedge \mathcal{S}_g(O,T) \wedge \neg\mathcal{H}_i(\_,T,\_,a|c)\\ &\Rightarrow \mathcal{S}_h(O,T)\end{aligned}$$

Note that condition C2 is trivially fulfilled since DRC relations do not have duplicates by definition.

Furthermore, we require the lemma presented below:

Lemma 1 (One Operation per Iteration and Object). *The DPI query guarantees that at most one operation per object is scheduled during each scheduler iteration:*

$$\begin{aligned}&\forall i,T,O,A : \mathcal{H}^E_i(\_,T,\_,A,O) \wedge A = r|w \wedge (T \neq T_2 \vee A \neq A_2)\\ &\Rightarrow \neg\mathcal{H}^E_i(\_,T_2,\_,A_2,O)\end{aligned}$$

respectively

$$\begin{aligned}&\forall i,T,O,A : \mathcal{H}^E_i(\_,T,\_,A,O) \wedge A = r|w \wedge \mathcal{H}^E_i(\_,T_2,\_,A_2,O)\\ &\Rightarrow T = T_2 \wedge A = A_2\end{aligned}$$

Proof. Lemma 1 follows from the definition of *MinStmtPerObj* in the definition of *Schedule*. We prove the claim by contradiction.

Assumption of contradiction: We assume the opposite (negation) of the condition holds for some T, O, A and i:

$$\begin{aligned}&\mathcal{H}^E_i(\_,T,\_,A,O) \wedge A = r|w \wedge (T \neq T_2 \vee A \neq A_2)\wedge\\ &\mathcal{H}^E_i(\_,T_2,\_,A_2,O) \qquad (1)\end{aligned}$$

From the definition of $\mathcal{H}^E_i, \mathcal{H}_i, \mathcal{E}_i$ and *Schedule*, we can deduce that

$$\begin{aligned}\mathcal{H}^E_i &= \mathcal{H}_i - \mathcal{H}_{i-1}\\ &= \mathcal{H}_{i-1} \cup \mathcal{E}_i - \mathcal{H}_{i-1}\\ &= \mathcal{E}_i\\ &= \mathit{Schedule}(\mathcal{R}_i,\mathcal{H}_{i-1})\\ &=\{\mathit{GenId}(),T,N,A,O|\mathcal{R}_i(T,N,A,O)\wedge\\ &\quad(\mathit{MinStmtPerObj}(\_,T) \vee (A=a \vee A=c))\}\end{aligned}$$

Which means the following equivalence holds for $\mathcal{H}^E_i$

$$\begin{aligned}&\forall T,N,A,O : \mathcal{H}^E_i(\_,T,N,A,O)\\ &\Leftrightarrow\mathcal{R}_i(T,N,A,O) \wedge (\mathit{MinStmtPerObj}(\_,T)\vee\\ &(A=a \vee A=c))\end{aligned}$$

Using this equivalence to replace $\mathcal{H}^E_i$ in equation 1 we get:

$$\begin{aligned}&\mathcal{H}^E_i(\_,T,\_,A,O) \wedge A = r|w \wedge (T \neq T_2 \vee A \neq A_2)\wedge\\ &\mathcal{H}^E_i(\_,T_2,\_,A_2,O)\\ &\Leftrightarrow \mathcal{R}_i(T,\_,A,O) \wedge (\mathit{MinStmtPerObj}(\_,T)\vee\\ &(A=a \vee A=c)) \wedge A = r|w \wedge (T \neq T_2 \vee A \neq A_2)\wedge\\ &\mathcal{R}_i(T_2,\_,A_2,O) \wedge (\mathit{MinStmtPerObj}(\_,T_2)\vee\\ &(A_2=a \vee A_2=c))\end{aligned}$$

From $\mathcal{R}_i(T,\_,A,O) \wedge A = r|w$ follows $(A=a \vee A=c) = \mathit{false}$. And from $\mathcal{R}_i(T,\_,A,O)\wedge A = r|w$ and $\mathcal{R}_i(T_2,\_,A_2,O)$ we can deduce $(A_2 = a \vee A_2 = c) = \mathit{false}$ because $O = \mathit{Null}$ if $A_2 = a|c$ and this contradicts with $A = r|w$.

$$\begin{aligned}&\Leftrightarrow \mathcal{R}_i(T,\_,A,O) \wedge \mathit{MinStmtPerObj}(\_,T) \wedge A = r|w\wedge\\ &\mathcal{R}_i(T_2,\_,A_2,O) \wedge \mathit{MinStmtPerObj}(\_,T_2)\wedge\\ &(T \neq T_2 \vee A \neq A_2)\end{aligned}$$

This implies:

$$\mathit{MinStmtPerObj}(O,T) \wedge \mathit{MinStmtPerObj}(O,T_2)$$

If $T = T_2$, then the contradiction follows from $P1$. If $T \neq T_2$, the contradiction follows from the semantics of grouping and aggregation, because aggregation produces a single result tuple for each group by value (O). □

Theorem 1 (Correctness of the SS2PL DPI). *Each sequence of states of relation $\mathcal{H}$ $<\mathcal{H}_0,\ldots>$ generated with an arbitrary input sequence $<New_0,\ldots>$ according to Definition 1 respects the SS2PL protocol as defined by Definition 2.*

PROOF. Theorem 1.
Given the recursive definition of the request database states, we prove our initial claim by induction over the number i of scheduler iterations and show that each state of relation $\mathcal{H}_i$ conforms to the SS2PL protocol.

Therefore, we first prove that $\mathcal{H}_i$ with $i = 0$ conforms to the SS2PL protocol. Assuming that $\mathcal{H}_i$ for scheduler iterations $i = x$ is correct, we proof the correctness of $\mathcal{H}_i$ for $i = x + 1$. Thus, all states of relation $\mathcal{H}_i$ conform to the SS2PL protocol, i.e., conditions $C1$ to $C6$ hold for all states of relation $\mathcal{H}_i$.

The proofs are done by transforming C1-C6 using $\mathcal{X}$, $\mathcal{S}$, P1 and P2 and by showing that C1-C6 evaluate to true for the corresponding request database state. This approach does not only apply for proving the correctness of the SS2PL DPI but also to the DPIs of other scheduling protocols.

Induction start

From the definition of the initial state of the request database we know that $\mathcal{H}_0 = \mathcal{E}_0 = \mathcal{R}_0 = New_0 = \emptyset$. In the proofs for each condition $C1$ to $C6$, we can omit universal quantified variables that range over the number of scheduler iterations, since for all request database relations version 0 is the only one that is defined at scheduler iteration 0.

Conditions C1-C6 trivially evaluate to true since by definition $\mathcal{H}_0 = \mathcal{E}_0 = \mathcal{R}_0 = New_0 = \mathcal{H}_0^E = \emptyset$.
Furthermore, $\mathcal{S}_i = \mathcal{X}_i = \emptyset$ for $i \leq 1$:

$$\begin{aligned}\mathcal{X}_1 =& \{O,T|\mathcal{H}_0(\_,T,\_,w,O) \wedge \neg\mathcal{H}_0(\_,T,\_,a|c,\_)\}\\ =& \{O,T|false \wedge true\}\\ =& \{O,T|false\}\\ =& \emptyset\\ \mathcal{S}_1 =& \{O,T|\mathcal{H}_0(\_,T,\_,r,O) \wedge \neg\mathcal{H}_0(\_,T,\_,w,O) \wedge\\ & \neg\mathcal{H}_0(\_,T,\_,a|c,\_))\}\\ =& \{O,T|false \wedge true \wedge true)\}\\ =& \{O,T|false)\}\\ =& \emptyset\end{aligned}$$

Condition C1:

$$\begin{aligned}&\forall T,O : \mathcal{H}_0^E(\_,T,\_,r,O) \Rightarrow \mathcal{S}_1(O,T) \vee \mathcal{X}_1(O,T)\\ \wedge&\forall T,O : \mathcal{H}_0^E(\_,T,\_,w,O) \Rightarrow \mathcal{X}_1(O,T)\end{aligned}$$

$\mathcal{H}^E, \mathcal{S}, \mathcal{X}$ evaluate to false since $\mathcal{H}_i^E = \emptyset$ for $i \leq 0$ and $\mathcal{S}_i = \mathcal{X}_i = \emptyset$ for $i \leq 1$ by definition

$$\begin{aligned}&\Leftrightarrow \forall T,O : false \Rightarrow false \vee false\\ &\quad \wedge \forall T,O : false \Rightarrow false\\ &\Leftrightarrow true\end{aligned}$$

Condition C3:

$$\begin{aligned}&\forall O,T,T_2 : \mathcal{X}_1(O,T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_1(O,T_2)\\ \wedge&\forall O,T,T_2 : \mathcal{X}_1(O,T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_1(O,T_2)\\ \wedge&\forall O,T,T_2 : \mathcal{S}_1(O,T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_1(O,T_2)\end{aligned}$$

$\mathcal{S}, \mathcal{X}$ evaluate to false since $\mathcal{S}_i = \mathcal{X}_i = \emptyset$ for $i \leq 1$ by definition

$$\begin{aligned}&\Leftrightarrow \forall O,T,T_2 : false \wedge T \neq T_2 \Rightarrow true\\ &\quad \wedge \forall O,T,T_2 : false \wedge T \neq T_2 \Rightarrow true\\ &\quad \wedge \forall O,T,T_2 : false \wedge T \neq T_2 \Rightarrow true\\ &\Leftrightarrow true\end{aligned}$$

Condition C5:

$$\forall T,h : h < 1 \wedge \mathcal{H}_h(\_,T,\_,a|c,\_) \Rightarrow \neg\mathcal{X}_1(\_,T) \wedge \neg\mathcal{S}_1(\_,T)$$

$\mathcal{H}, \mathcal{S}, \mathcal{X}$ evaluate to false since $\mathcal{H}_i = \emptyset$ for $i \leq 0$ and $\mathcal{S}_i = \mathcal{X}_i = \emptyset$ for $i \leq 1$ by definition

$$\begin{aligned}&\Leftrightarrow \forall T,h : h < 0 \wedge false \Rightarrow true \wedge true\\ &\Leftrightarrow true\end{aligned}$$

Condition C6:

$$\begin{aligned}&\forall T,O,g,h : g \leq h \leq 1 \wedge \mathcal{X}_g(O,T) \wedge \neg\mathcal{H}_0(\_,T,\_,a|c)\\ &\quad \Rightarrow \mathcal{X}_h(O,T)\\ \wedge&\forall T,O,g,h : g \leq h \leq 1 \wedge \mathcal{S}_g(O,T) \wedge \neg\mathcal{H}_0(\_,T,\_,a|c)\\ &\quad \Rightarrow \mathcal{S}_h(O,T)\end{aligned}$$

$\mathcal{H}, \mathcal{S}, \mathcal{X}$ evaluate to false since $\mathcal{H}_i = \emptyset$ for $i \leq 0$ and $\mathcal{S}_i = \mathcal{X}_i = \emptyset$ for $i \leq 1$ by definition

$$\begin{aligned}&\Leftrightarrow \forall T,O,g,h : g \leq h \leq 1 \wedge false \wedge true \Rightarrow false\\ &\quad \wedge\ \forall T,O,g,h : g \leq h \leq 1 \wedge false \wedge true \Rightarrow false\\ &\Leftrightarrow true\end{aligned}$$

Induction step

We already proved that conditions $C1$ to $C6$ hold for relation $\mathcal{H}_i$ with $i = 0$. Assuming that conditions $C1$ to $C6$ hold for $\mathcal{H}_i$ with $i = x$, we now show that these conditions hold for $\mathcal{H}_i$ with $i = x + 1$ too. Thus, conditions $C1$ to $C6$ hold for states of relation $\mathcal{H}$.

Condition C1:

$$\begin{aligned}&\forall T,O,i : \mathcal{H}_i^E(\_,T,\_,r,O) \Rightarrow \mathcal{S}_{i+1}(O,T) \vee \mathcal{X}_{i+1}(O,T)\\ \wedge&\forall T,O,i : \mathcal{H}_i^E(\_,T,\_,w,O) \Rightarrow \mathcal{X}_{i+1}(O,T)\end{aligned}$$

We set $i = x + 1$ and remove the universal quantification on i.

$$\begin{aligned}&\forall T,O : \mathcal{H}_{x+1}^E(\_,T,\_,r,O) \Rightarrow \mathcal{S}_{x+2}(O,T) \vee \mathcal{X}_{x+2}(O,T)\\ \wedge&\forall T,O : \mathcal{H}_{x+1}^E(\_,T,\_,w,O) \Rightarrow \mathcal{X}_{x+2}(O,T)\end{aligned}$$

Proving first universal quantification of $C1$. First, we prove the first of the two universal quantifications of $C1$.

$$\forall T,O : \mathcal{H}_{x+1}^E(\_,T,\_,r,O) \Rightarrow \mathcal{S}_{x+2}(O,T) \vee \mathcal{X}_{x+2}(O,T)$$

(S,X) Replace $\mathcal{S}$ and $\mathcal{X}$ by their definition

$$\begin{aligned}&\Leftrightarrow \forall T,O : \mathcal{H}_{x+1}^E(\_,T,\_,r,O) \Rightarrow\\ &\quad (\mathcal{H}_{x+1}(\_,T,\_,r,O) \wedge \neg\mathcal{H}_{x+1}(\_,T,\_,w,O) \wedge\\ &\quad \neg\mathcal{H}_{x+1}(\_,T,\_,a|c,\_))\\ &\quad \vee (\mathcal{H}_{x+1}(\_,T,\_,w,O) \wedge \neg\mathcal{H}_{x+1}(\_,T,\_,a|c,\_))\end{aligned}$$

To prove that this part of the condition holds we distinguish two cases. Either $H_{x+1}(\_, T, \_, w, o)$ holds (case 1) or $\neg H_{x+1}(\_, T, \_, w, o)$ holds (case 2). Assume case 1 holds:

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \Rightarrow \\
&\quad (\mathcal{H}_{x+1}(I, T, \_, r, O) \wedge \mathit{false} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \\
&\quad \vee (\mathit{true} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \\
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \Rightarrow \mathit{false} \vee \\
&\quad (\mathit{true} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \\
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \Rightarrow \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)
\end{aligned}$$

If $\neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ holds then also $\neg(\mathcal{H}^E_u(\_, T, \_, a|c, \_) \wedge u \leq x+1)$ holds by $\mathcal{H}_i \subseteq \mathcal{H}_{i+1}$ which follows from the recurisve definition of H:

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \\
&\quad \Rightarrow \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge \\
&\quad \forall T, O, u : \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \\
&\quad \Rightarrow \neg(\mathcal{H}^E_u(\_, T, \_, a|c, \_) \wedge u \leq x+1) \\
&\text{(Trans) By transitivity} \\
&\Leftrightarrow \forall T, O, u : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \\
&\quad \Rightarrow \neg(\mathcal{H}^E_u(\_, T, \_, a|c, \_) \wedge u \leq x+1) \\
&\text{(Rewr) Rewrite implication} \\
&\Leftrightarrow \forall T, O, u : \neg\mathcal{H}^E_{x+1}(\_, T, \_, r, O) \vee \\
&\quad \neg(\mathcal{H}^E_u(\_, T, \_, a|c, \_) \wedge u \leq x+1) \\
&\text{(De Morgan) Apply De Morgan's law, reordering} \\
&\Leftrightarrow \forall T, O, u : \neg(\mathcal{H}^E_{x+1}(\_, T, \_, r, O) \wedge u \leq x+1) \vee \\
&\quad \neg\mathcal{H}^E_u(\_, T, \_, a|c, \_) \\
&\text{(Rewr) Rewrite implication} \\
&\Leftrightarrow \forall T, O, u : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \wedge u \leq x+1 \\
&\quad \Rightarrow \neg\mathcal{H}^E_u(\_, T, \_, a|c, \_)
\end{aligned}$$

Now we have to distinguish two cases. Either $u = x+1$ holds (case 1a) or $u < x+1$ holds (case 1b). Assuming case 1a holds, both requests, $\mathcal{R}_x(\_, T, \_, r, O)$ and $\mathcal{R}_{x+1}(\_, T, \_, a|c, \_)$, would have been in relation $\mathcal{R}_{x+1}$ which contradicts precondition P1. Thus, $\neg\mathcal{H}^E_u(\_, T, \_, a|c, \_)$ and the implication evaluate to true. Assuming case 1b holds, the implication is fulfilled by definition of precondition P2.
Assume case 2 holds:

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \Rightarrow \mathcal{H}_{x+1}(I, T, \_, r, O) \wedge \\
&\quad \mathit{true} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \\
&\quad \vee (\mathit{false} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \\
&\text{(Del) Delete irrelevant terms} \\
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \Rightarrow \mathcal{H}_{x+1}(I, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)
\end{aligned}$$

A tuple of $\mathcal{H}^E_{x+1}$ is always a tuple of $\mathcal{H}_{x+1}$ by definition of $\mathcal{H}^E$ ($\mathcal{H}^E_{x+1} \subseteq \mathcal{H}_{x+1}$):

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, r, O) \\
&\quad \Rightarrow \mathit{true} \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)
\end{aligned}$$

$\neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ evaluates to true for the same reasons as in cases 1a and 1b

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(I, T, \_, r, O) \Rightarrow \mathit{true} \\
&\Leftrightarrow \mathit{true}
\end{aligned}$$

Proving the second universal quantification of $C1$. Now we prove the second quantification of the condition C1.

$$\begin{aligned}
&\forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, w, O) \Rightarrow \mathcal{X}_{x+2}(O, T) \\
&\text{(X) Substitute } \mathcal{X} \text{ by its definition} \\
&\Leftrightarrow \forall T, O : \mathcal{H}^E_{x+1}(\_, T, \_, w, O) \\
&\quad \Rightarrow \mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)\}
\end{aligned}$$

If we rewrite the implication $a \to (b \wedge \neg c)$ to $(a \to b) \wedge (a \to \neg c)$ we get:

$$\begin{aligned}
&\Leftrightarrow \forall T, O : (\mathcal{H}^E_{x+1}(\_, T, \_, w, O) \Rightarrow \mathcal{H}_{x+1}(\_, T, \_, w, O)) \\
&\quad \wedge (\mathcal{H}^E_{x+1}(\_, T, \_, w, O) \Rightarrow \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_))
\end{aligned}$$

A tuple of $\mathcal{H}^E_{x+1}$ is always a tuple of $\mathcal{H}_{x+1}$ by definition of $\mathcal{H}^E$ ($\mathcal{H}^E_{x+1} \subseteq \mathcal{H}_{x+1}$).

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathit{true} \wedge (\mathcal{H}^E_{x+1}(\_, T, \_, w, O) \\
&\quad \Rightarrow \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_))
\end{aligned}$$

If $\mathcal{H}^E_{x+1}(\_, T, \_, w, O)$, no $\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ may exist for the same reasons as in cases 1a and 1b

$$\begin{aligned}
&\Leftrightarrow \forall T, O : \mathit{true} \wedge \mathit{true} \\
&\Leftrightarrow \mathit{true}
\end{aligned}$$

Condition C3:

$$\begin{aligned}
&\forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2) \\
&\wedge \forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{i+1}(O, T_2) \\
&\wedge \forall O, T, T_2, i : \mathcal{S}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2)
\end{aligned}$$

We set $i = x+1$ and prove the three universal quantified sub-expressions of $C3$ separately by contradiction.

$$\begin{aligned}
&\forall O, T, T_2 : \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{x+2}(O, T_2) \\
&\wedge \forall O, T, T_2 : \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{x+2}(O, T_2) \\
&\wedge \forall O, T, T_2 : \mathcal{S}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{x+2}(O, T_2)
\end{aligned}$$

Proving the first universal quantification of $C3$. We prove the first universal quantification of $C3$ by contradiction.

Assumption of contradiction: We assume the opposite (negation) of the first quantification holds.

$$\begin{aligned}
&\forall O, T, T_2 : \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{x+2}(O, T_2) \\
&\text{(Neg) Negation} \\
&\Leftrightarrow \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \wedge \mathcal{X}_{x+2}(O, T_2) \\
&\text{(X) Substitute } \mathcal{X} \text{ by its definition} \\
&\Leftrightarrow \mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge T \neq T_2 \\
&\quad \wedge \mathcal{H}_{x+1}(\_, T_2, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)
\end{aligned}$$

From Lemma 1 we know that the two operations on O have to have been scheduled during different scheduler iterations

h and i, i for the operation executed by T and h for the operation of transaction T_2 with either $h < i$ or $i < h$. Since the use of T and T_2 in the condition is symmetric, let wlog $h < i$. Thus, we can deduce that $i = x + 1$.

$$\Leftrightarrow h < x+1 \wedge \mathcal{H}^E_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge$$
$$T \neq T_2 \wedge \mathcal{H}^E_h(\_, T_2, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)$$

From $\neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)$, $h < x + 1$, and the definition of $\mathcal{X}$, we follow that $\mathcal{X}_{x+1}(O, T_2)$ holds. Based on the recursive definition of $\mathcal{H}$, we know that $\mathcal{H}^E_{x+1}(\_, T, \_w, O)$ implies $\mathit{LegalOps}_{x+1}(T, N, O)$ which in turn implies $\neg\mathit{OpsOnXLO}_{x+1}(T, N, O) \wedge \neg\mathit{WOpsOnSLO}_{x+1}(T, N, O)$, i.e., $\mathcal{H}^E_{x+1}$ can only contain the tuple $\mathcal{H}^E_{x+1}(\_, T, \_w, O)$ if this tuple belongs to $\mathit{LegalOps}_{x+1}$. This stands in contradiction with $\mathit{OpsOnXLO}_{x+1}(T, N, O)$ which follows from $\mathcal{X}_{x+1}(O, T_2)$.

Proving second universal quantification of $C3$. We proceed by proving the second universal quantification of $C3$ by contradiction:

$$\forall O, T, T_2 : \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{x+2}(O, T_2)$$

(X) Substitute $\mathcal{X}$ by its definition and rewrite implication

$$\Leftrightarrow \forall O, T, T_2 : \neg(\mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge$$
$$\neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge T \neq T_2)$$
$$\vee \neg(\mathcal{H}_{x+1}(I, T_2, \_, r, O) \wedge \neg\mathcal{H}_{x+1}(\_, T_2, \_, w, O) \wedge$$
$$\neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)$$

Assumption of contradiction: We assume the opposite (negation) of the quantification holds.

$$\Leftrightarrow \mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge T \neq T_2$$
$$\wedge \mathcal{H}_{x+1}(I, T_2, \_, r, O) \wedge \neg\mathcal{H}_{x+1}(\_, T_2, \_, w, O) \wedge$$
$$\neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)$$

Using the same argument as for the first quantification in $C3$, we can deduce that the read and write operation on O have been executed during different scheduler runs, scheduler run i, for the operation executed by T, and scheduler run h, for the operation executed by $T2$, with either $h < i, i = x + 1$ or $i < h, h = x + 1$.

Asume case $h = x + 1$ holds:

$$\Leftrightarrow i < x+1 \wedge \mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_) \wedge T \neq T_2$$
$$\wedge \mathcal{H}_{x+1}(I, T_2, \_, r, O) \wedge \neg\mathcal{H}_{x+1}(\_, T_2, \_, w, O) \wedge$$
$$\neg\mathcal{H}_{x+1}(\_, T_2, \_, a|c, \_)$$

We know that $X_{x+1}(O, T)$ and the contradiction follows from

$$\mathcal{H}_{x+1}(I, T_2, \_, r, O) \Rightarrow \mathit{LegalOps}_{x+1}(T_2, N, O)$$
$$\Rightarrow \neg\mathit{OpsOnXLO}_{x+1}(T_2, N, O)$$
$$\overset{\lightning}{\Leftrightarrow} \mathit{OpsOnXLO}_{x+1}(T_2, N, O) \Leftarrow X_{x+1}(O, T)$$

Asume case $i = x + 1$ holds:

$$\Leftrightarrow h < x+1 \wedge \mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_) \wedge$$
$$T \neq T_2 \wedge \mathcal{H}_h(I, T_2, \_, r, O) \wedge \neg\mathcal{H}_h(\_, T_2, \_, w, O) \wedge$$
$$\neg\mathcal{H}_h(\_, T_2, \_, a|c, \_)$$

If $i = x+1$, we know that $\mathcal{S}_{x+1}(O, T_2)$ and the contradiction follows from:

$$\mathcal{H}_{x+1}(I, T, \_, w, O) \Rightarrow \mathit{LegalOps}_{x+1}(T, N, O)$$
$$\Rightarrow \neg\mathit{WOpsOnSLO}_{x+1}(T, N, O)$$
$$\overset{\lightning}{\Leftrightarrow} \mathit{WOpsOnSLO}_{x+1}(T_2, N, O) \Leftarrow \mathcal{S}_{x+1}(O, T_2)$$

Proving third universal quantification of $C3$. Proving the third part of $C3$ is redundant since the proof of the second part applies for the third part as well due to equivalence of both parts:

$$\forall O, T, T_2 : \mathcal{S}_{x+2}(O, T) \wedge T \neq T_2$$
$$\Rightarrow \neg\mathcal{X}_{x+2}(O, T_2)$$

(Rewrite) Rewrite implication

$$\Leftrightarrow \forall O, T, T_2 : \neg(\mathcal{S}_{x+2}(O, T) \wedge T \neq T_2) \vee$$
$$\neg\mathcal{X}_{x+2}(O, T_2)$$

(De Morgan) Apply De Morgan's law

$$\Leftrightarrow \forall O, T, T_2 : \neg\mathcal{S}_{x+2}(O, T) \vee \neg(T \neq T_2) \vee$$
$$\neg\mathcal{X}_{x+2}(O, T_2)$$

(Order) Rearranging

$$\Leftrightarrow \forall O, T, T_2 : \neg\mathcal{X}_{x+2}(O, T_2) \vee \neg(T \neq T_2) \vee$$
$$\neg\mathcal{S}_{x+2}(O, T)$$

(De Morgan) Apply De Morgan's law

$$\Leftrightarrow \forall O, T, T_2 : \neg(\mathcal{X}_{x+2}(O, T_2) \wedge T \neq T_2) \vee$$
$$\neg\mathcal{S}_{x+2}(O, T)$$

(Rewrite) Rewrite for implication

$$\Leftrightarrow \forall O, T, T_2 : \mathcal{X}_{x+2}(O, T_2) \wedge T \neq T_2$$
$$\Rightarrow \neg\mathcal{S}_{x+2}(O, T)$$

Renaming the variables in the formula we establish the equivalence with the second part:

$$\Leftrightarrow \forall O, T, T_2 : \mathcal{X}_{x+2}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{x+2}(O, T_2)$$

Condition C5:

$$\forall T, h, i : h < i+1 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_)$$
$$\Rightarrow \neg\mathcal{X}_{i+1}(\_, T) \wedge \neg\mathcal{S}_{i+1}(\_, T)$$

In condition $C5$, we replace i with $x + 1$.

$$\forall T, h : h < x+2 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_)$$
$$\Rightarrow \neg\mathcal{X}_{x+2}(\_, T) \wedge \neg\mathcal{S}_{x+2}(\_, T)$$

$(\mathcal{X}, \mathcal{S})$ Substitute $\mathcal{X}$ and $\mathcal{S}$ by their definition

$$\Leftrightarrow \forall T, h : h < x+2 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow$$
$$\neg(\mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \wedge$$
$$\neg(\mathcal{H}_{x+1}(I, T, \_, r, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \wedge$$
$$\neg\mathcal{H}_{x+1}(\_, T, \_, a|c, \_))$$

(De Morgan) Apply De Morgan's law

$$\Leftrightarrow \forall T, h : h < x+2 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow$$
$$(\neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \vee \mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \wedge$$
$$(\neg\mathcal{H}_{x+1}(I, T, \_, r, O) \vee \mathcal{H}_{x+1}(\_, T, \_, w, O) \vee$$
$$\mathcal{H}_{x+1}(\_, T, \_, a|c, \_))$$

If $\mathcal{H}_h(\_, T, \_, a|c, \_)$ then also $\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ with $h < x+1$ by $\mathcal{H}_i \subseteq \mathcal{H}_{i+1}$ which follows from the recursive definition of $\mathcal{H}$.

$$\begin{aligned}
&\Leftrightarrow \forall T, h : h < x+1 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow \\
&\quad (\neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \vee true) \wedge \\
&\quad (\neg\mathcal{H}_{x+1}(I, T, \_, r, O) \vee \mathcal{H}_{x+1}(\_, T, \_, w, O) \vee true) \\
&\Leftrightarrow true
\end{aligned}$$

Condition C6:

$$\begin{aligned}
&\forall T, O, g, h, i : g \le h \le i+1 \wedge \mathcal{X}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{X}_h(O, T) \\
&\wedge \forall T, O, g, h, i : g \le h \le i+1 \wedge \mathcal{S}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{S}_h(O, T)
\end{aligned}$$

Proving first universal quantification of C6. We replace the i with $x+1$ and remove the universal quantification on i.

$$\begin{aligned}
&\forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{X}_g(O, T) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{X}_h(O, T)
\end{aligned}$$

We replace $\mathcal{X}$ by its definition.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, a|c, \_) \\
&\quad \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, w, O) \wedge \\
&\quad \neg\mathcal{H}_{h-1}(\_, T, \_, a|c, \_)
\end{aligned}$$

From $\neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \wedge g \le h \le x+2$ we deduce that $\neg\mathcal{H}_{h-1}(\_, T, \_, a|c, \_)$ and $\neg\mathcal{H}_{g-1}(\_, T, \_, a|c, \_)$ hold, because the highest value $g-1$ and $h-1$ can have is $x+1$.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge true \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, w, O) \wedge true \\
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, w, O)
\end{aligned}$$

If $\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge g \le h \le x+2$ holds then $\mathcal{H}_{h-1}(\_, T, \_, w, O)$ evaluates to true by $\mathcal{H}_i \subseteq \mathcal{H}_{i+1}$ which follows from the recursive definition of $\mathcal{H}$.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow true \\
&\Leftrightarrow true
\end{aligned}$$

Proving second universal quantification of C6. We replace the i with $x+1$ and remove the universal quantification on i.

$$\begin{aligned}
&\forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{S}_g(O, T) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{S}_h(O, T)
\end{aligned}$$

At first, we replace $\mathcal{S}$ by its definition.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{g-1}(\_, T, \_, a|c, \_) \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{h-1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{h-1}(\_, T, \_, a|c, \_)
\end{aligned}$$

From $\neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \wedge g \le h \le x+2$ we deduce that $\neg\mathcal{H}_{h-1}(\_, T, \_, a|c, \_)$ and $\neg\mathcal{H}_{g-1}(\_, T, \_, a|c, \_)$ hold, because the highest value $g-1$ and $h-1$ can have is $x+1$.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge true \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, r, O) \wedge \neg\mathcal{H}_{h-1}(\_, T, \_, w, O) \wedge true \\
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow \mathcal{H}_{h-1}(\_, T, \_, r, O) \wedge \neg\mathcal{H}_{h-1}(\_, T, \_, w, O) \wedge
\end{aligned}$$

If $\mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge g \le h \le x+2$ holds then $\mathcal{H}_{h-1}(\_, T, \_, r, O)$ evaluates to true by $\mathcal{H}_i \subseteq \mathcal{H}_{i+1}$.

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow true \wedge \neg\mathcal{H}_{h-1}(\_, T, \_, w, O) \\
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge \\
&\quad \neg\mathcal{H}_{g-1}(\_, T, \_, w, O) \wedge \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \\
&\quad \Rightarrow \neg\mathcal{H}_{h-1}(\_, T, \_, w, O)
\end{aligned}$$

Now we have to distinguish two cases. Either $\neg\mathcal{H}_{h-1}(\_, T, \_, w, O)$ holds (case 1) or $\mathcal{H}_{h-1}(\_, T, \_, w, O)$ holds (case 2).

Case 1, assuming $\neg\mathcal{H}_{h-1}(\_, T, \_, w, O)$ holds, then also $\neg\mathcal{H}_{g-1}(\_, T, \_, w, O)$ holds by $\mathcal{H}_i \subseteq \mathcal{H}_{i+1}$ which follows from the recursive definition of $\mathcal{H}$

$$\begin{aligned}
&\Leftrightarrow \forall T, O, g, h : g \le h \le x+2 \wedge \mathcal{H}_{g-1}(\_, T, \_, r, O) \wedge true \wedge \\
&\quad \neg\mathcal{H}_{x+1}(\_, T, \_, a|c) \Rightarrow true \\
&\Leftrightarrow true
\end{aligned}$$

Case 2, assuming $\mathcal{H}_{h-1}(\_, T, \_, w, O)$ holds, then $\mathcal{X}_{h-1}(O, T)$ holds. And because of condition C3, transaction T may not hold a shared lock while holding an exclusive lock. Thus, we follow that the lock hold by T changed from a shared to an exclusive lock and the first universal quantification of C6 hold, which we have proved before. □

We proved that conditions C1-C6 hold for each state of relation $\mathcal{H}$. For simplicity, we did not yet consider the *Irrelevant* query, which deletes statements from relation $\mathcal{H}$ which are irrelevant for scheduling decisions. Now, we have to show that the execution of *Irrelevant* has no influence on the correctness of relation $\mathcal{H}$. Therefore, we have to adapt Definition 1 to Definition 3.

Note that the step to remove revoked requests (*Revoked* query) from relation $\mathcal{R}$ as explained in Section 1.1 does not influence the correctness of *Schedule*. This is because Revoked() deletes requests which have not been scheduled and does not touch information of relation $\mathcal{H}$.

DEFINITION 3 (REQUEST DATABASE STATE'). *For an input sequence $<New_0, \ldots, New_i>$, $i \in N$, we call the tuple $RDB_i = (\mathcal{H}_i, \mathcal{R}_i, \mathcal{E}_i, New_i)$ the request database state after scheduler iteration i. RDB_i is defined recursively:*

$$\mathcal{H}_i = \mathcal{E}_i = \mathcal{R}_i = New_i = \emptyset, for\ i \le 0$$

$$\begin{aligned}
&\mathcal{R}_{i+1} = \mathcal{R}_i \cup New_{i+1} - \{T, N, A, O \mid \mathcal{E}_i(\_, T, N, A, O)\} \\
&\mathcal{E}_{i+1} = Schedule(\mathcal{R}_{i+1}, \mathcal{H}_i) \\
&\mathcal{T}_{i+1} = \mathcal{H}_i \cup \mathcal{E}_{i+1} \\
&\mathcal{H}_{i+1} = \mathcal{T}_{i+1} - Irrelevant(\mathcal{T}_{i+1})
\end{aligned}$$

(Thereby, $\mathcal{T}_i$ denotes an auxiliary set to define $\mathcal{H}$)

THEOREM 2 (EQUIVALENCE OF DEFINITION 1 AND 3). *Each sequence of states $<\mathcal{H}_0, \dots>$ of relation $\mathcal{H}$ generated with an arbitrary input sequence $<New_0, \dots>$ according to Definition 1 is equal to a generation according to Definition 3.*

PROOF. Theorem 2.
Relation $\mathcal{H}$ gets changed by (a) the insertion of the statements selected by *Schedule* and (b) the deletion of statements of unfinished transactions by *Irrelevant*. We have to show two facts: (1) *Irrelevant* does not influence the set of requests selected by *Schedule*. (2) Conditions C1-C6 hold for the states of relation $\mathcal{H}$ despite of the execution of *Irrelevant*.

Fact 1: In *Schedule*, only the selection of read and write operations is based on information of relation $\mathcal{H}$, more precisely $\mathcal{S}$ and $\mathcal{X}$. $\mathcal{S}$ and $\mathcal{X}$ only consider statements of non-finished transactions. And *Irrelevant* only deletes statements of already finished transactions:

$$Irrelevant = \{I, T, N, A, O \mid \mathcal{H}(I, T, N, A, O) \wedge \mathcal{H}(\_, T, \_, a|c, \_)\}$$

We prove that *Irrelevant* executed in scheduler iteration i does not change $\mathcal{X}_{i+1}$ by contradiction.

$$\mathcal{X}_{i+1} = \{O, T | \mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)\}$$

I.e. the following rule holds for $\mathcal{X}$:

$$\forall O, T, i : \mathcal{X}_{i+1}(O, T) \Rightarrow \mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)$$

Assumption of contradiction: We assume that a transaction lost a write lock after *Irrelevant* has been executed.

Since by definition *Irrelevant* only deletes tuples, it has to have deleted a write statement. But also by definition, *Irrelevant* only deletes statements if $\mathcal{H}_i(\_, T, \_, a|c, \_)$ holds. Thus, the contradiction follows from:

$$\neg\mathcal{H}_i(\_, T, \_, a|c, \_) \overset{\lightning}{\Leftrightarrow} \mathcal{H}_i(\_, T, \_, a|c, \_)\}$$

We prove that *Irrelevant* executed in scheduler iteration i does not change $\mathcal{S}_{i+1}$ by contradiction.

$$\mathcal{S}_{i+1} = \{O, T | \mathcal{H}_i(\_, T, \_, r, O) \wedge \neg\mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)\}$$

I.e. the following rule holds for $\mathcal{X}$:

$$\forall O, T, i : \mathcal{S}_{i+1}(O, T) \Rightarrow \mathcal{H}_i(\_, T, \_, r, O) \wedge \neg\mathcal{H}_i(\_, T, \_, w, O) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c, \_)$$

Assumption of contradiction: We assume that a transaction lost a read lock after *Irrelevant* has been executed.

Since by definition *Irrelevant* only deletes tuples, it has to have deleted a read statement. But since by definition, *Irrelevant* only deletes statements if $\mathcal{H}_i(\_, T, \_, a|c, \_)$ holds. Thus, the contradiction follows from:

$$\neg\mathcal{H}_i(\_, T, \_, a|c, \_) \overset{\lightning}{\Leftrightarrow} \mathcal{H}_i(\_, T, \_, a|c, \_)$$

From this we follow that Fact 1 holds.

Fact 2: Now we have to show that *Irrelevant* does not violate any proof of conditions C1-C6.

Condition C1:

$$\forall T, O, i : \mathcal{H}_i^E(\_, T, \_, r, O) \Rightarrow \mathcal{S}_{i+1}(O, T) \vee \mathcal{X}_{i+1}(O, T)$$
$$\wedge \forall T, O, i : \mathcal{H}_i^E(\_, T, \_, w, O) \Rightarrow \mathcal{X}_{i+1}(O, T)$$

Assume condition C1 holds for some i after Definition 1. We have to show that after deleting the requests from *Irrelevant* this condition still holds. If *Irrelevant* deletes any request of a transaction T from the history, it deletes all requests of T. Therefore, if *Irrelevant* deletes all reads of transaction T on object O (changing the evaluation of the left hand side of the implication to false: $\mathcal{H}_i(\_, T, \_, r, O)$), then also the right hand side of the implication evaluates to false, because $\mathcal{S}_{i+1}(O, T)$ requires $\mathcal{H}_i(\_, T, \_, r, O)$ or $\mathcal{H}_i(\_, T, \_, w, O)$ and $\mathcal{X}_{i+1}(O, T)$ requires $\mathcal{H}_i(\_, T, \_, w, O)$. If *Irrelevant* deletes writes of transaction T on object O (changing the evaluation of the left hand side of the implication of the second universal quantification to false: $\mathcal{H}_i(\_, T, \_, w, O)$), then also the right hand side of the implication evaluates to false, because $\mathcal{X}_{i+1}(O, T)$ requires $\mathcal{H}_i(\_, T, \_, w, O)$. Controversially, if the deletions of *Irrelevant* cause the right hand side to evaluate to true, then we can use the same argument to show that the left hand side does too.

Condition C3:

$$\forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2)$$
$$\wedge \forall O, T, T_2, i : \mathcal{X}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{S}_{i+1}(O, T_2)$$
$$\wedge \forall O, T, T_2, i : \mathcal{S}_{i+1}(O, T) \wedge T \neq T_2 \Rightarrow \neg\mathcal{X}_{i+1}(O, T_2)$$

Assume condition C3 holds for some i after Definition 1. We have to show that after deleting the requests from *Irrelevant* this condition still holds. Assume *Irrelevant* deletes all or non requests a transaction T from the history, the right side of the implications do not change. The lefts sides of the implications can only change to false. Hence, the case $true \Rightarrow false$ is not possible.

Condition C5:

$$\forall T, h, i : h < i + 1 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_)$$
$$\Rightarrow \neg\mathcal{X}_{i+1}(\_, T) \wedge \neg\mathcal{S}_{i+1}(\_, T)$$

Assume condition C5 holds for some i after Definition 1. We have to show this condition still holds after Definition 3. If *Irrelevant* deletes any request of a transaction T from the history, it deletes all requests of T. Therefore, if *Irrelevant* deletes an abort or commit statement of transaction T (changing the evaluation of the left hand side of the implication to false), then the right hand side of the implication does not change. This is because *Irrelevant* solely deletes requests and due to P3. Thus, C5 evaluates to true.

In the induction step for condition C5, we stated that if $\mathcal{H}_h(\_, T, \_, a|c, \_) \wedge h < x + 1$ holds then also $\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ holds, as shown in the following excerpt of the proof of condition C5:

$$\Leftrightarrow \forall T, h : h < x + 2 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow$$
$$(\neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \vee \mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \wedge$$
$$(\neg\mathcal{H}_{x+1}(I, T, \_, r, O) \vee \mathcal{H}_{x+1}(\_, T, \_, w, O) \vee$$
$$\mathcal{H}_{x+1}(\_, T, \_, a|c, \_))$$

If $\mathcal{H}_h(\_, T, \_, a|c, \_)$ then also $\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ with $h < x+1$:

$$\begin{aligned}&\Leftrightarrow \forall T, h : h < x+1 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow\\ &\quad (\neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \vee true) \wedge\\ &\quad (\neg\mathcal{H}_{x+1}(I, T, \_, r, O) \vee \mathcal{H}_{x+1}(\_, T, \_, w, O) \vee true)\\ &\Leftrightarrow true\end{aligned}$$

But even if *Irrelevant* deletes the abort resp. commit statement and $\mathcal{H}_{x+1}(\_, T, \_, a|c, \_)$ evaluates to false, this does not influence the result of the proof because *Irrelevant* always deletes all tuples of a finished transaction by definition. Thus, the proof evaluates to true as follows:

$$\begin{aligned}&\Leftrightarrow \forall T, h : h < x+2 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow\\ &\quad (\neg\mathcal{H}_{x+1}(\_, T, \_, w, O) \vee \mathcal{H}_{x+1}(\_, T, \_, a|c, \_)) \wedge\\ &\quad (\neg\mathcal{H}_{x+1}(I, T, \_, r, O) \vee \mathcal{H}_{x+1}(\_, T, \_, w, O) \vee\\ &\quad \mathcal{H}_{x+1}(\_, T, \_, a|c, \_))\\ &\Leftrightarrow \forall T, h : h < x+1 \wedge \mathcal{H}_h(\_, T, \_, a|c, \_) \Rightarrow\\ &\quad (true \vee false) \wedge (true \vee false \vee false)\\ &\Leftrightarrow true\end{aligned}$$

Condition C6:

$$\begin{aligned}&\forall T, O, g, h, i : g \leq h \leq i+1 \wedge \mathcal{X}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c)\\ &\quad \Rightarrow \mathcal{X}_h(O, T)\\ &\wedge \forall T, O, g, h, i : g \leq h \leq i+1 \wedge \mathcal{S}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c)\\ &\quad \Rightarrow \mathcal{S}_h(O, T)\end{aligned}$$

Assume condition C6 holds for some input sequence after Definition 1. We have to show that C6 holds too after Definition 3. We prove this fact by contradiction.

Assumption of contradition: Condition C6 breaks for an i. I.e., the opposite holds.

$$\begin{aligned}&(g \leq h \leq i+1 \wedge \mathcal{X}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c) \wedge \neg\mathcal{X}_h(O, T))\\ &\vee (g \leq h \leq i+1 \wedge \mathcal{S}_g(O, T) \wedge \neg\mathcal{H}_i(\_, T, \_, a|c) \wedge \neg\mathcal{S}_h(O, T))\end{aligned}$$

We have to distinguish two cases. Either the first part evaluates to true or the second one.

First part: The first part evaluates to true if $\mathcal{X}_g(O, T)$ holds for some O and T and $\neg\mathcal{H}_i(\_, T, \_, a|c)$ and $\neg\mathcal{X}_h(O, T)$ hold as well. This is not possible because, firstly, this contradicts to C6 and, secondly, *Irrelevant* always deletes all tuples of T by definition, i.e., $\mathcal{X}_g(O, T)$ would not hold after an execution of *Irrelevant*.

The second part is analog.

Thus, Fact 2 holds and *Irrelevant* does not influence the correctness of relation $\mathcal{H}$ and *Schedule* always selects the same set of statements independent of the execution of *Irrelevant*. $\Box$

3. REFERENCES

[1] C. Tilgner. Declarative Scheduling in Highly Scalable Systems. In *EDBT '10: Proceedings of the 2010 EDBT/ICDT Workshops*, pages 1–6. ACM, March 2010.

[2] G. Weikum and G. Vossen. *Transactional Information Systems.* Morgan Kaufmann Publishers, 2002.

RDB_0

New

TA	Seq	Op	Ob

R

TA	Seq	Op	Ob

E

ID	TA	Seq	Op	Ob

H

ID	TA	Seq	Op	Ob

S

Ob	TA

X

Ob	TA

RDB_1

New

TA	Seq	Op	Ob
15	1	R	A
8	1	W	B

R

TA	Seq	Op	Ob
15	1	R	A
8	1	W	B

E

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B

S

Ob	TA

X

Ob	TA

RDB_2

New

TA	Seq	Op	Ob
8	2	C	-
15	2	W	A
14	1	R	D

R

TA	Seq	Op	Ob
8	2	C	-
15	2	W	A
14	1	R	D

E

ID	TA	Seq	Op	Ob
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D

S

Ob	TA
A	15

X

Ob	TA
B	8

RDB_3

New

TA	Seq	Op	Ob
21	1	R	A
27	1	W	A
34	1	W	B
32	1	W	D
2	1	R	C

R

TA	Seq	Op	Ob
21	1	R	A
27	1	W	A
34	1	W	B
32	1	W	D
2	1	R	C

E

ID	TA	Seq	Op	Ob
6	34	1	W	B
7	2	1	R	C

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D
6	34	1	W	B
7	2	1	R	C

S

Ob	TA
D	14

X

Ob	TA
A	15

RDB_4

New

TA	Seq	Op	Ob
15	3	C	-

R

TA	Seq	Op	Ob
21	1	R	A
27	1	W	A
32	1	W	D
15	3	C	-

E

ID	TA	Seq	Op	Ob
8	15	3	C	-

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D
6	34	1	W	B
7	2	1	R	C
8	15	3	C	-

S

Ob	TA
D	14
C	2

X

Ob	TA
A	15
B	34

RDB_5

New

TA	Seq	Op	Ob

R

TA	Seq	Op	Ob
21	1	R	A
27	1	W	A
32	1	W	D

E

ID	TA	Seq	Op	Ob
9	21	1	R	A

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D
6	34	1	W	B
7	2	1	R	C
8	15	3	C	-
9	21	1	R	A

S

Ob	TA
D	14
C	2

X

Ob	TA
B	34

RDB_6

New

TA	Seq	Op	Ob
14	2	C	-
21	2	C	-

R

TA	Seq	Op	Ob
27	1	W	A
32	1	W	D
14	2	C	-
21	2	C	-

E

ID	TA	Seq	Op	Ob
10	14	2	C	-
11	21	2	C	-

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D
6	34	1	W	B
7	2	1	R	C
8	15	3	C	-
9	21	1	R	A
10	14	2	C	-
11	21	2	C	-

S

Ob	TA
D	14
C	2
A	21

X

Ob	TA
B	34

RDB_7

New

TA	Seq	Op	Ob
7	1	R	E

R

TA	Seq	Op	Ob
27	1	W	A
32	1	W	D
7	1	R	E

E

ID	TA	Seq	Op	Ob
12	27	1	W	A
13	32	1	W	D
14	7	1	R	E

H

ID	TA	Seq	Op	Ob
1	15	1	R	A
2	8	1	W	B
3	8	2	C	-
4	15	2	W	A
5	14	1	R	D
6	34	1	W	B
7	2	1	R	C
8	15	3	C	-
9	21	1	R	A
10	14	2	C	-
11	21	2	C	-
12	27	1	W	A
13	32	1	W	D
14	7	1	R	E

S

Ob	TA
C	2

X

Ob	TA
B	34

Figure 4: Example Illustrating Request Database States (RDB)