Algebraic Specification of Serializability for Partitioned Transactions

Walter Hussak and John Keane

Abstract—The usual correctness condition for a schedule of concurrent database transactions is some form of serializability of the transactions. For general forms, the problem of deciding whether a schedule is serializable is NP-complete. In those cases other approaches to proving correctness, using proof rules that allow the steps of the proof of serializability to be guided manually, are desirable. Such an approach is possible in the case of conflict serializability which is proved algorithmically by deriving serial schedules using commutativity of non-conflicting operations. However, conflict serializability can be an unnecessarily strong form of serializability restricting concurrency and thereby reducing performance. In practice, weaker, more general, forms of serializability for extended models of transactions are used. Currently, there are no known methods using proof rules for proving those general forms of serializability. In this paper, we define serializability for an extended model of partitioned transactions, which we show to be as expressive as serializability for general partitioned transactions. An algebraic method for proving general serializability is obtained by giving an initial-algebra specification of serializable schedules of concurrent transactions in the model. This demonstrates that it is possible to conduct algebraic proofs of correctness of concurrent transactions in general cases.

Keywords—Algebraic Specification, Partitioned Transactions, Serializability.

I. INTRODUCTION

The standard database transaction model [4] has proved to be inadequate when applied practically, for example to transactions of long duration. Requiring serializability of such transactions results in long duration waits and abortion of long transactions. A proposed solution to overcome these problems has been to partition transactions. Models of partitioned transactions and suggested application areas have been given in [16], [11] and [5]. These allow an increase in concurrency for executing transactions whilst maintaining database consistency based on data consistency conditions weaker than standard serializability. However, data consistency conditions are useful only if there are suitable methods for proving them. In the standard model of transactions, this means proving serializability. The proof techniques are algorithms based on dependency graphs [12] and [17]. These algorithms are NP-complete and for this reason stronger forms of serializability that can be proved in polynomial time, notably ‘conflict’ serializability [12], have been suggested. Interestingly, conflict serializability can also be characterized algorithmically - a schedule is serializable if and only if it can be transformed into a serial schedule by commuting non-conflicting operations (i.e. two reads or two operations accessing different data items). Such commutativity-based forms of serializability have also been characterized in temporal logic [14] and [8]. However, there have been almost no algebraic or proof-rule characterizations of forms of serializability that are not commutativity-based. The specification in [13] is in a temporal logic without an evident axiomatization and although modest extensions beyond commutativity-based serializability are possible by the techniques of [10] and [8], no proof-theoretic method that uses inference rules has been formulated for general non-commutative serializability. It should be noted that any specification of serializability in propositional temporal logic has a proof method by virtue of an axiomatization of the logic and, as it is decidable, a fully automatic algorithm. However, such an algorithm cannot improve on one based on dependency graphs as such logics are PSPACE-complete [18].

In this paper, we give an algebraic method for proving general serializability. There are two main results. Firstly, an extended partitioned transaction model is given and is shown to be as expressive as the general partitioned transaction model. Secondly, an algebraic method is given for proving general serializability in our extended model. Although general serializability is not commutative, we are able to give an initial-algebra specification of serializable schedules in the style of [6] making use of conditional equations to overcome the problem. The paper is structured as follows. Section II gives our extended model of partitioned transactions. In Section III it is proved to be as expressive as the general partitioned transaction model. The algebraic specification of serializable schedules for our extended transaction model is in Section IV. The conclusions are in Section V.

II. A MODEL OF PARTITIONED TRANSACTIONS

A. Steps, transactions and histories

A read or write step to \( r(i, x) \) or \( w(i, x) \)

respectively. A more condensed notation will be used in places in Section IV where

\[ r_i^T \]

will replace \( r(i, x) \) and \( w(i, x) \). The \( i \) subscript or \( x \) superscript will also be omitted occasionally when of no interest. A transaction \( (T_i; \leq_i) \) is a finite partially ordered set of steps with transaction identifier \( i \). The corresponding irreflexive order is denoted \( < \). A transaction system is a finite set of transactions.

A schedule or history \( h \) is a sequence of steps. The total order of the sequence is denoted \( \leq_h \) with \( <_h \) as the irreflexive
version. If $T = \{T_1, \ldots, T_n\}$ is a transaction system, then $h$ is a history of $T$ if the steps of $h$ are the steps of all the transactions in $T$ placed in some order such that steps that are ordered in each $T_i$ preserve that order in $h$, that is $s_1 <_i s_2 \Rightarrow s_1 <_h s_2$

A history of $T$ represents an execution of the transactions in $T$ in an interleaved fashion.

As an example, suppose that a transaction $T_i$ reads an integer data item $a$, updates $a$, reads an integer data item $b$, and updates $b$. Then, $T_i$ is seen as just a succession of formal reads and writes to $a$ and $b$ thus:

$$r(1,a)w(1,a)r(1,b)w(1,b)$$

without further information about the nature of $a$ or $b$ or the read or write steps. If there are two such transactions $T_1$ and $T_2$, then the following sequence of steps is a history:

$$r(1,a)w(1,a)r(2,a)w(2,a)r(2,b)w(2,b)r(1,b)w(1,b)$$

B. Partitions and tiers

Let $T_P = \{T_1, \ldots, T_n\}$ be a transaction system. Then a partition of $T$ is a collection of partitions of the individual $T_i$'s

$$\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$$

where $\mathcal{P}_i = \{T_{i1}, \ldots, T_{ip_i}\}$ is a partition of the set of steps of $T_i$ ($1 \leq i \leq n$). If we wish to indicate that a step belongs to $T_{ij}$ in the partition $\mathcal{P}_i$, we shall add an extra (first) argument as in:

$$r(j,i,x) \text{ or } w(j,i,x)$$

Each member $T_{ij}$ of a partition $\mathcal{P}_i$ is called a $\mathcal{P}_i$-segment (or just segment). If, in addition, for each $i$, $q$, and $r$ ($q$ and $r$ distinct) it is impossible to have steps $s_1$, $s_2$, $s_3$, and $s_4$ such that

$$s_1, s_2 \in T_{iq}, \quad s_3, s_4 \in T_{ir}$$

then $\mathcal{P}$ is a tiered partition. Each segment is then called a tier, and a transaction partitioned into tiers will be called a tiered transaction. We shall use the tiered transaction model as our model of partitioned transactions. Tiers have a high intuitive appeal. By condition (1), a transaction that is partitioned into tiers can be pictured as being constructed of a succession of tiers placed end to end without overlap. This gives the tiered partitioned model simple algebraic properties. An example of partitioned transactions will be given in II.D below.

C. History equivalence and tiered-serializability

Two histories are 'equivalent' iff they have the same set of read and write steps, and every read step 'sees' (i.e. reads) the same value in both and also the final database state is the same for both. In detail, let $h$ be any history $s_1 \ldots s_K$ (not necessarily of any particular transaction system). A read step $s_l \leq_h s_k$

and there is no write step $s$ to $x$ such that $s_l \leq_h s \leq_h s_k$

A history $h'$ is equivalent to $h$, denoted $h' \sim h$, iff there is a permutation $\rho$ of $\{1, \ldots, K\}$ such that $h'$ is $s_{\rho(1)} \cdots s_{\rho(K)}$ and

$$\text{sees}_{h}(s_k) = s_l \Leftrightarrow \text{sees}_{h'}(s_k) = s_l$$

for every read step $s_k$ and write step $s_l$ and, furthermore, for each variable $x$, $h$ and $h'$ have the same last write steps.

A history of a transaction system $T$ is serial iff given (distinct) transactions $T_{i1}$, $T_{i2}$ in $T$ either

$$s_1 <_h s_2 \text{ for all } s_1 \in T_{i1}, s_2 \in T_{i2}$$

or

$$s_2 <_h s_1 \text{ for all } s_1 \in T_{i1}, s_2 \in T_{i2}$$

and serializable iff it is equivalent to a serial history. If the transactions are partitioned by a tiered partition $\mathcal{P}_i$, then a history of $T$ is $\mathcal{P}_i$-serial or just tiered-serial (with the understanding that the tiers are as in $\mathcal{P}_i$) iff given distinct transactions $T_{i1}$ and $T_{i2}$ in $T$, and tiers $T_{i1j}$, $T_{i2j}$, either

$$s_1 <_h s_2 \text{ for all } s_1 \in T_{i1j1}, s_2 \in T_{i2j2}$$

or

$$s_2 <_h s_1 \text{ for all } s_1 \in T_{i1j1}, s_2 \in T_{i2j2}$$

Clearly, a serial history is tiered-serial. A history $h$ is $\mathcal{P}_i$-serializable or just tiered-serializable iff it is equivalent to a tiered-serial history.

D. An example of tiered-serializability

The reason for tiered serializability is seen from the following example. Suppose that a travel database has data items $x$ and $y$ corresponding to booking information for flights between between destinations $A$ and $B$ and flights between destinations $B$ and $C$ respectively. Further, suppose that transactions $T_i$ arrange journeys between destinations $A$ and $C$, by first obtaining flight booking information and booking flights between $A$ and $B$ and then obtaining flight information and booking connecting flights between $B$ and $C$, so that each $T_i$ comprises the following succession of read and write steps:

$$r(1,x)w(1,x)r(1,y)w(1,y)$$

Consider the following history for the concurrent execution of three such transactions $T_1$, $T_2$ and $T_3$:

$$r(1,x)w(1,x)r(2,x)w(2,x)r(2,y)w(2,y)$$

$$r(1,y)r(3,x)w(1,y)r(3,x)w(3,y)r(3,y)w(3,y)$$

In the history (2), $T_1$ and $T_2$ are not serialized because $T_1$ books its first flight and then $T_2$ books both its first and
connecting flight before \( T_1 \) manages to book its connecting flight - if all of \( T_1 \) had been executed before \( T_2 \), \( T_1 \) would, in general, have had more connecting flights to choose from and may have booked those differently; if all of \( T_2 \) had been executed before \( T_1 \), \( T_2 \) would, in general, have had more first flights to choose from and may have booked those differently. However, the history (2) still books first and connecting flights consistently and allowing it increases concurrency. Here, consistency means tiered-serializability where the accesses to first flight and connecting flight information respectively constitute the tiers, i.e. each \( T_i \) is the tiered transaction:

\[
\begin{align*}
&\text{(1) becomes the tiered-serializable history:} \\
&\ r(1, i, x)w(1, i, x)r(2, i, y)w(2, i, y)
\end{align*}
\]

and (2) becomes the tiered-serializable history:

\[
\begin{align*}
&\text{(2) becomes the tiered-serializable history:} \\
&\ r(1, 1, x)w(1, 1, x)r(1, 2, x)w(1, 2, x)r(2, 2, y)w(2, 2, y)
\end{align*}
\]

\[
\begin{align*}
&\text{equivalent to the tiered-serial history:} \\
&\ r(1, 1, x)w(1, 1, x)r(1, 2, x)w(1, 2, x)r(2, 2, y)w(2, 2, y)
\end{align*}
\]

\[
\begin{align*}
&\text{as swapping the places of } r(1, 3, x) \text{ and } w(2, 1, y) \text{ does not affect the sees function. Notice that serializability of at least the tiers is needed as, for example, the history} \\
&\ r(1, 1, x)r(1, 2, x)w(1, 1, x)r(1, 2, x)r(2, 2, y)w(2, 2, y)
\end{align*}
\]

\[
\begin{align*}
&\text{can result in an inconsistent database state, as flights booked by transaction } T_1 \text{ may be overwritten by those booked by } T_2. \\
&\text{Also, notice that tiers cannot be treated as separate transactions because of interdependencies between tiers belonging to a common transaction - in the example just given, the particular connecting flights booked depend on the particular first flights booked for reasons such as connections being able to be made.}
\end{align*}
\]

### III. Expressiveness

In Section II above, we defined \( \mathcal{P}_i \)-serializability for tiered partitions \( \mathcal{P}_i \). A similar definition of \( \mathcal{P} \)-serializability (see below) can be given for arbitrary partitions \( \mathcal{P} \) that do not satisfy condition (1), and corresponds to serializability for general models of partitioned transactions such as that given in [16]. We have chosen tiers because they can be specified algebraically. In this section we show that they are also as expressive, in that the set of \( \mathcal{P} \)-serial histories for an arbitrary partition \( \mathcal{P} \) is equal to the set of \( \mathcal{P}_i \)-serial histories for some tiered partition \( \mathcal{P}_i \). As such, a required data consistency condition that is defined in terms of \( \mathcal{P} \)-serializability for some partition \( \mathcal{P} \) can, with a little extra effort, be defined in terms of \( \mathcal{P}_i \)-serializability for some tiered partition \( \mathcal{P}_i \).

#### B. Slices

Let \( h \) be a history. Then, by choosing steps at which changes of transaction occur in the history, we can find steps

\[
\begin{align*}
&\text{such that:} \\
&(i) \text{ for } 1 \leq i \leq m, \text{ all steps in } \{s : s_{2i-1} \leq s \leq s_{2i}\} \text{ belong to the same transaction;} \\
&(ii) \text{ for } 1 \leq i \leq m, \text{ all steps in } \{s : s_{2i-1} \leq s \leq s_{2i}\} \text{ and } \{s : s_{2i+1} \leq s \leq s_{2i+2}\} \text{ belong to different transactions;}
\end{align*}
\]

The subsequence of \( h \) comprising the steps in \( \{s : s_{2i-1} \leq s \leq s_{2i}\} \) is called the \( i \)-th slice of \( h \) (\( 1 \leq i \leq m \)). Slices are used in the following convenient characterization of \( \mathcal{P} \)-serial histories:

**Lemma 1** Let \( T \) be a transaction system and \( \mathcal{P} \) a partition of \( T \). A history \( h \) of \( T \) is \( \mathcal{P} \)-serial if and only if each slice of \( h \) is a union of complete \( \mathcal{P} \)-segments of the same transaction.

**Proof** If a segment \( P_1 \) intersects two distinct slices, there will be an intermediate slice with steps belonging to a segment \( P_2 \) of some other transaction. Thus, there will be \( s_{11}, s_{12} \in P_1 \) and \( s_2 \in P_2 \) such that

\[
s_{11} < h < s_2 < h < s_{12}
\]

Thus, \( h \) is not \( \mathcal{P} \)-serial by the definition in III.A. This proves the “only if” part of the lemma. The “if” part is obvious.

#### C. Tier-definability

A set of histories \( \mathcal{H} \) of a transaction system \( T \) is \( \mathcal{P} \)-definable iff there is a tiered partition \( \mathcal{P}_T \) of \( T \) such that \( \mathcal{H} \cong \mathcal{P}_T \)-definable. The next theorem shows that general \( \mathcal{P} \)-serial histories can be generated by (a suitable choice of) tiers. This proves the generality of the tiered model.

**Theorem 2** Let \( T \) be a transaction system and \( \mathcal{P} \) a partition of \( T \). A set of histories \( \mathcal{H} \) of \( T \) that is \( \mathcal{P} \)-definable is also \( \mathcal{P}_T \)-definable.
Proof Suppose that \( H \) is a set of histories that \( \mathcal{P}_1 \)-definable. Define the relation \( \sim_i \), on a transaction \( T_i \in T \) to be such that \( s_1 \sim_i s_2 \) iff, for all \( h \in H \),
\[
s_1 \leq_h s \leq_h s_2 \lor s_2 \leq_h s \leq_h s_1 \Rightarrow s \in T_i,
\]
i.e. \( s_1 \) and \( s_2 \) cannot be separated by a step \( s \) not in \( T_i \) in any history \( h \in H \). Firstly, we check that \( \sim_i \) is an equivalence relation on \( T_i \). It is trivially reflexive and symmetric. Suppose that \( s_1 \sim_i s_2 \) and \( s_2 \sim_i s_3 \). Let \( h \in H \) and \( s \) be such that \( s_1 \leq_h s \leq_h s_3 \). If \( s_2 \leq_h s \), then \( s_2 \leq_h s \leq_h s_3 \) and so \( s \in T_i \) as \( s_2 \sim_i s_3 \). If \( s < h s_2 \), then \( s_1 \leq_h s \leq_h s_2 \) and so \( s \in T_i \) as \( s_1 \sim_i s_2 \). By a symmetric argument \( s_3 \leq_h s \leq_h s_1 \Rightarrow s_1 \in T_i \). It follows that \( \sim_i \) is transitive and thus an equivalence relation.

Next, we show that the equivalence classes define a tiered partition of \( T \). Let \( T_{iq} \) and \( T_{ir} \) be distinct equivalence classes and assume, on the contrary, that condition (1) of II.B is not satisfied for \( T_{iq} \) and \( T_{ir} \). Then, there are \( s_1, s_2 \in T_{iq}, s_3, s_4 \in T_{ir} \) such that
\[
s_1 \leq s_1 \leq h s_3, \quad s_4 \leq s_2 \leq h s_2
\]
If \( s \leq s_2 \) then
\[
s_1 \leq s_1 \leq h s_2
\]
and so \( s \neq_1 s_4 \), contradicting the fact that \( s_1 \) and \( s_2 \) belong to the same equivalence class \( T_{iq} \). On the other hand, if \( s_2 \leq s_1 \) then
\[
s_4 \leq s_2 \leq h s_2
\]
as \( s_4 \neq s_2 \) implies \( s_1 \leq s_2 \). This means that \( s_4 \neq s_3 \), contrary to \( s_3 \) and \( s_4 \) being in the same equivalence class \( T_{ir} \). We have thus shown that the equivalence classes define a tiered partition, \( \mathcal{P}_1 \), say, of \( T \).

Finally, it remains to show that \( H \) is \( \mathcal{P}_1 \)-definable. Now, as \( H \) is \( \mathcal{P} \)-definable, it consists of the \( \mathcal{P} \)-serial histories of \( T \). If \( s_1 \) and \( s_2 \) belong to the same \( \mathcal{P} \)-segment of \( T_i \) then, given a history \( h \in H \), as \( h \) is \( \mathcal{P} \)-serial, it is easy to see from the definition of \( \mathcal{P} \)-serial histories in III.A that no step \( s \notin T_i \) can come between \( s_1 \) and \( s_2 \) in \( h \). Therefore, by the definition of \( \sim_i \), \( s_1 \sim_i s_2 \) and so \( s_1 \) and \( s_2 \) are in the same tier. Thus, each tier is a union of complete \( \mathcal{P} \)-segments. Also, given \( h \in H \), two steps \( s_1 \) and \( s_2 \) in the same tier, of transaction \( T_1 \) say, cannot be in different slices otherwise, by the definition of slices in III.B, they could be separated by some \( s \notin T_i \), which would contradict the fact that, being in the same tier, \( s_1 \sim_i s_2 \). Therefore, the slices of each \( h \in H \) are unions of complete tiers. It follows that
\[
h \in H \Leftrightarrow h \text{ is } \mathcal{P} \text{-serial}
\]
\[
\Leftrightarrow \text{the slices of } h \text{ are a union of complete } \mathcal{P} \text{-segments} \text{ by Lemma 1}
\]
\[
\Leftrightarrow \text{the slices of } h \text{ are a union of complete tiers}
\]
\[
\Leftrightarrow h \text{ is } \mathcal{P}_1 \text{-serial} \text{ by Lemma 1}
\]

IV. Algebraic Theory

The basis of the algebraic approach is the observation that sometimes an equivalent history is produced if two adjacent steps are switched. Thus
\[
r_1^x r_2^y w_3^x w_3^y \text{ and } r_1^x r_2^y r_3^y w_3^x (x \neq y, i \neq j)
\]
are equivalent. For general serializability, not all serializable histories can be derived from serial histories using this form of commutativity. The problem is with commuting write steps. For example,
\[
r_1^x w_2^x r_2^x w_3^x (x \neq y, i \neq j, j \neq k, i \neq k)
\]
the two writes \( w_1^x \) and \( w_1^y \) may be commuted to form an equivalent history, whereas equivalence will not be preserved if they are commuted in
\[
r_1^x r_2^x w_3^x r_2^x w_3^x (x \neq y, i \neq j, j \neq k, i \neq k)
\]
as \( r_k^x \) will, in general, read a different value of \( x \).

The solution we shall adopt in such cases is essentially to commute ‘write blocks’ where each write block comprises a write step followed by the read steps that read that write. Thus, the last history is equivalent to
\[
r_1^x w_2^x r_2^x w_3^x \quad \text{where the write blocks } w_1^x \text{ and } w_1^y \text{ have been commuted.}
\]

The theory we present enables all tiered-serializable histories to be derived from tiered-serial histories. In IV.A, we give an equational presentation of the theory in the style of [6]. In IV.B, we show that the algebra of tiered-serializable histories is ‘initial’ for this presentation, which means that histories defined to be tiered-serializable as in II.C are exactly those that can be proved to be tiered-serializable by the equations in IV.A.

A. Equational presentation

In the theory below, the operation \( \tau \) and \( \omega \) generate read and write steps \( \tau(j, i, x) \) and \( \omega(j, i, x) \) where \( j, i \) and \( x \) are natural number labels for tiers, transactions and data items respectively. Natural numbers are specified by the operations \( 1, \text{ succ } \) and \( \text{ is equal } \). Histories are formed by use of the associative infix operator \( . \) to concatenate other histories. For example, the following expression:
\[
\tau(1, \text{ succ}(1), \text{ succ}(\text{ succ}(1))), \omega(1, \text{ succ}(1), \text{ succ}(\text{ succ}(1)))
\]
represents the history comprising a read step followed by a write step to a data item labelled 3 by a tier labelled 1 of a transaction labelled 2. The predicate operations \( \text{ tier } \), \( \text{ trans } \) and \( \text{ ts } \) have the following meanings: \( \text{ tier } (h,j,i) \) asserts that \( h \) is a \( j \)-labelled tier of an \( i \)-labelled transaction, \( \text{ trans } (h,m,i) \) asserts that \( h \) is an \( i \)-labelled transaction with \( m \) tiers, and \( \text{ ts } (h,n) \) asserts that \( h \) is a tiered-serial history with \( n \) transactions.

Equations (E1)-(E4) belong to the specification of natural numbers, and (E5) is associativity of the \( . \) concatenation operator. Equations (E6)-(E13) enable all serial histories of transactions to be generated, where the constituent transactions
are labelled by consecutive integers starting at 1, and where
the tiers within each transaction are also labelled consecu-
tively. The equation (E14) allows other tiered-serial histories,
where transactions and tiers are not necessarily labelled by
consecutive integers, to be generated by commuting adjacent
tiers belonging to different transactions (tiers $h_3$ and $h_4$
is commuted in (E14)).

Proving serializability for a history $h$ requires proving $tserz(h) = \text{TRUE}$. The predicate $tserz$ and history equiv-
alence predicates $Tequiv$ and $Vequiv$ along with correspond-
ing equations (E15)-(E34) are discussed in IV.B.

SORTS

$\text{N}$ (naturals), $\text{H}$ (histories)

VARIABLES

$m,n,i,j,1,2,\ldots,j_1,j_2,\ldots,x,y : \text{N}$

$h,h_1,h_2,\ldots : \text{H}$

OPERATIONS

(O1) $1 : \rightarrow \text{N}$

(O2) $\text{succ} : \text{N} \rightarrow \text{N}$

(O3) $r : \text{N},\text{N},\text{N} \rightarrow \text{H}$

(O4) $w : \text{N},\text{N},\text{N} \rightarrow \text{H}$

(O5) $._{\_isequal} : \text{N},\text{N} \rightarrow \text{B}$

(O6) $._{\_} : \text{H},\text{H} \rightarrow \text{B}$

(O7) $\text{tier} : \text{H},\text{N},\text{N} \rightarrow \text{B}$

(O8) $\text{trans} : \text{H},\text{N},\text{N} \rightarrow \text{B}$

(O9) $\text{tser} : \text{H},\text{N} \rightarrow \text{B}$

(O10) $\text{wb} : \text{H},\text{N} \rightarrow \text{B}$

(O11) $\text{tequiv} : \text{H},\text{H} \rightarrow \text{B}$

(O12) $\text{vequiv} : \text{H},\text{H} \rightarrow \text{B}$

(EQUATIONS)

(E1) $1 \_isequal 1 = \text{TRUE}$

(E2) $1 \_isequal \text{succ}(n) = \text{FALSE}$

(E3) $\text{m isequal n} = n \_isequal m$

(E4) $\text{m isequal succ}(m) \_isequal \text{succ}(m)$

(E5) $(h_1.h_2).h_3 = h_1(h_2.h_3)$

(E6) $\text{tier}(r(j,i,x),j,i) = \text{TRUE}$

(E7) $\text{tier}(w(j,i,x),j,i) = \text{TRUE}$

(E8) $\text{tier}(h,r(j,i,x),j,i) = \text{TRUE}$

(E9) $\text{tier}(h.w(j,i,x),j,i) = \text{TRUE}$

(E10) $\text{trans}(h,1,i) = \text{TRUE}$

(E11) $\text{trans}(h_1.h_2,\text{succ}(m),i) = \text{TRUE}$

(E12) $\text{tser}(h,1) = \text{TRUE}$

(E13) $\text{tser}(h_1.h_2,\text{succ}(m)) = \text{TRUE}$

(E14) $\text{tser}(h_1.h_2.h_4.h_3.h_5.h_6,n) = \text{TRUE}$

(E15) $\text{tequiv}(h,h) = \text{TRUE}$

(E16) $\text{tequiv}(h_1,h_2) = \text{tequiv}(h_2,h_1)$

(E17) $\text{tequiv}(h_1,h_3) = \text{TRUE}$

(E18) $\text{tequiv}(h_1.h_2,h_2) = \text{TRUE}$

(E19) $\text{tequiv}(h_1,h_2,h_2) = \text{TRUE}$

(E20) $\text{vequiv}(h,h) = \text{TRUE}$

(E21) $\text{vequiv}(h_1,h_2) = \text{vequiv}(h_2,h_1)$

(E22) $\text{vequiv}(h_1,h_3) = \text{TRUE}$

(E23) $\text{vequiv}(h_1,h_4,h_2) = \text{TRUE}$

(E24) $\text{vequiv}(h_1.h_2,h_4,h_2) = \text{TRUE}$

(E25) $\text{tequiv}(r(j_1,i_1,x).r(j_2,i_2,y),r(j_2,i_2,y).r(j_1,i_1,x)) = \text{TRUE}$

(E26) $\text{tequiv}(r(j_1,i_1,x).w(j_2,i_2,y),w(j_2,i_2,y).r(j_1,i_1,x)) = \text{TRUE}$

(E27) $\text{vequiv}(w(j_1,i_1,x).w(j_2,i_2,y),w(j_2,i_2,y).w(j_1,i_1,x)) = \text{TRUE}$

(E28) $\text{wb}(w(j,i,x),x) = \text{TRUE}$

(E29) $\text{wb}(h.r(j,i,x),x) = \text{TRUE}$

(E30) $\text{vequiv}(r(j_1,i_1,x).r(j_2,i_2,y),r(j_2,i_2,y).r(j_1,i_1,x)) = \text{TRUE}$

(E31) $\text{vequiv}(r(j_1,i_1,x).w(j_2,i_2,y),w(j_2,i_2,y).r(j_1,i_1,x)) = \text{TRUE}$

(E32) $\text{vequiv}(w(j_1,i_1,x).w(j_2,i_2,y),w(j_2,i_2,y).w(j_1,i_1,x)) = \text{TRUE}$

(E33) $\text{vequiv}(h_1.h_2.w(j,i,x),h_2.h_1.w(j,i,x)) = \text{TRUE}$

(E34) $\text{tserz}(h_1) = \text{TRUE}$

B. Initiality

The main result is Theorem 7 which shows that a history $h$ is tiered-serializable if and only if $tserz(h) = \text{TRUE}$
can be proved for the term $h$ corresponding to $h$. Proving$tserz(h) = \text{TRUE}$ requires proving $\text{tequiv}(h_1,h_2) = \text{TRUE}$
and $\text{vequiv}(h_1,h_2) = \text{TRUE}$ by (E34) for some history $h_1$. To this end, we first define the two new equivalences for
histories, namely ‘T-equivalence’ and ‘V-equivalence’. Which together amount to history equivalence defined in ILC, and show that they correspond to Tequiv and Vequiv in the equational presentation in IVA respectively. The proof that T-equivalence corresponds to Tequiv is immediate. The proof that V-equivalence corresponds to Vequiv is more difficult and is proved in Lemma 6, after defining a relation \( \approx^V \) on histories and proving intermediate results (Lemmas 3, 4 and 5) concerning \( \approx^V \) and V-equivalence.

Two histories \( h_1 \) and \( h_2 \) are T-equivalent iff they are histories of the same transaction system. It is easy to see that \( h_1 \) and \( h_2 \) are T-equivalent if and only if one can be obtained from the other by repeatedly commuting adjacent steps belonging to different transactions, i.e.

\[
h_1, h_2 \text{ are T-equivalent } \iff \text{Tequiv}(h_1, h_2) = \text{TRUE} \quad (3)
\]

where \( \text{Tequiv}(h_1, h_2) = \text{TRUE} \) can be proved for the terms \( h_1 \) and \( h_2 \) corresponding to \( h_1 \) and \( h_2 \) respectively, from the equations (E25) – (E27).

Two histories \( h_1 \) and \( h_2 \) comprising a write step to any given variable \( x \) iff it is of the form

\[
x_{i_1} \ldots x_{i_p}, \quad x_{j_1} \ldots x_{j_q}, \quad x_{k_1} \ldots x_{k_r}
\]

where \( s_i-1 \) is a write step to a variable \( x \) and \( n \) write blocks to \( x \) precede \( s_i-1 \) in \( h \). Then, given a permutation \( \pi \) of \( \{1, \ldots, n\} \), for

\[
h' = s_1 \ldots s_k \underbrace{x_p \ldots x_p}_{\text{w.block } \pi(1)} \ldots \underbrace{x_p \ldots x_p}_{\text{w.block } \pi(n)} s_{i-1} \ldots s_{i-1} \ldots s_K
\]

we have that \( h \approx^V h' \).

**Proof** The history \( h' \) can be obtained by repeatedly commuting adjacent write blocks, in the manner of (7), until the desired order of write blocks is achieved.

**Lemma 4** Let \( h \) be a history of the form

\[
h = s_1 \ldots s_k \underbrace{x_p \ldots x_p}_{\text{w.block } 1} \ldots \underbrace{x_p \ldots x_p}_{\text{w.block } n} s_{i-1} \ldots s_i \ldots s_K
\]

where \( s_{i-1} \) is a write step to a variable \( x \) and \( n \) write blocks to \( x \) precede \( s_{i-1} \) in \( h \). Then, given a permutation \( \pi \) of \( \{1, \ldots, n\} \), for

\[
h' = s_1 \ldots s_k \underbrace{x_p \ldots x_p}_{\text{w.block } \pi(1)} \ldots \underbrace{x_p \ldots x_p}_{\text{w.block } \pi(n)} s_{i-1} \ldots s_i \ldots s_K
\]

we have that \( h \approx^V h' \).

**Proof** The last write step to \( x \) in \( h \) cannot be moved to the left of another write step to \( x \) by use of any of (4), (5), (6) or (7).

**Lemma 5** Let \( h_1 \) and \( h_2 \) be histories. Then, \( h_1 \) and \( h_2 \) are V-equivalent if and only if \( h_1 \approx^V h_2 \).

**Proof** If \( h_1 \approx^V h_2 \), then \( h_1 \) and \( h_2 \) have the same sees function as (4)-(6) do not affect the write that a read step sees. Also, by Lemma 3, \( h_1 \) and \( h_2 \) have the same last write step to any given variable \( x \). Thus, \( h_1 \) and \( h_2 \) are V-equivalent. This proves the “if” part.

We now prove the “only if” part. Suppose that \( h_1 \) and \( h_2 \) are V-equivalent. By repeatedly commuting adjacent \( r^p \) and \( w^q \) steps in \( h_1 \) and \( h_2 \), as in (5), we can find histories \( h'_1 \) and \( h'_2 \) in write block form such that

\[
h_1 \approx^V h'_1, \quad h_2 \approx^V h'_2
\]

say

\[
h'_1 = w^{i_1} w^{i_2} \ldots w^{i_1} w^{i_2} \ldots w^{i_p} x_p \ldots x_p, \quad h'_2 = w^{j_1} w^{j_2} \ldots w^{j_1} w^{j_2} \ldots w^{j_q} x_q \ldots x_q
\]

Since \( h_1 \approx^V h'_1 \), by the “if” part of this lemma \( h_1 \) is V-equivalent to \( h'_1 \). Likewise, \( h_2 \) is V-equivalent to \( h'_2 \). As \( h_1 \) is V-equivalent to \( h_2 \) and V-equivalence is clearly transitive, it follows that \( h'_1 \) is V-equivalent to \( h'_2 \). From this it is clear that \( p = q \) and, indeed, \( h'_1 \) and \( h'_2 \) have the same write blocks though possibly in a different order. Furthermore, \( h'_1 \) and \( h'_2 \) have the same last write blocks to any given variable \( x \).

Applying (4), (5) and (6) repeatedly we can find \( h''_1 \) and \( h''_2 \) such that \( h_1 \approx^V h''_1, \quad h_2 \approx^V h''_2 \) and all write blocks to a given variable \( x \) are grouped together in \( h''_1 \) and \( h''_2 \). Let the two sequences of (distinct) variables corresponding to these groups of write blocks in \( h''_1 \) and \( h''_2 \) be:

\[
x_1^{(1)} \ldots x^{(1)}_p, \quad x_1^{(2)} \ldots x^{(2)}_p
\]
respectively. By further applying (4), (5) and (6) a multitude of times, effectively to commute adjacent whole groups of write blocks to different variables, we can find $h'''_1$ and $h'''_2$ such that $h'''_1 \approx_V h'''_2$ and the two sequences of (distinct) variables corresponding to the groups of write blocks in $h''_1$ and $h''_2$:

$$y_1^{(1)} \ldots y_p^{(1)} = y_1^{(2)} \ldots y_p^{(2)}$$

are the same for $h''_1$ and $h''_2$, i.e. $y_k^{(1)} = y_k^{(2)}$ ($1 \leq k \leq p$).

The only difference between $h'''_1$ and $h'''_2$ is the order in which write blocks occur within a write block grouping corresponding to a variable $x$. As noted above, $h'_1$ and $h'_2$ have the same last write blocks to $x$. It is clear from this and the construction of $h'''_1$, $h'''_2$, $h''_1$ and $h''_2$, that $h'''_1$ and $h'''_2$ have the same last write blocks to any given $x$. The other write blocks to this $h'''_1$ are just a permutation, say, of the remaining write blocks to $x$ in $h''_2$. By repeated use of Lemma 4 for each variable $x$ in turn we deduce that $h'''_1 \approx_V h''_2$.

To summarize,

$$h_1 \equiv_V h'_1 \approx_V h''_1 \approx_V h'''_1 \approx_V h'''_2 \approx_V h''_2 \approx_V h_2$$

It follows that $h_1 \approx_V h_2$.

Lemma 6 Let $h_1$ and $h_2$ be histories. Then, $h_1$ and $h_2$ are $V$-equivalent if and only if if $Vequiv(h_1,h_2) = TRUE$ can be proved for the terms $h_1$ and $h_2$ corresponding to $h_1$ and $h_2$ respectively.

Proof Follows from (8) and Lemma 5.

Theorem 7 A history is tiered-serializable if and only if $tserz(h)$=true can be proved for the term $h$ corresponding to $h$.

Proof

- $h$ tiered-serializable $\iff$ $h$ equivalent to tiered-serial history $h_1$
- $h_1$ tiered-serial, $h$ and $h_1$ T-equivalent and $V$-equivalent
- $h_1$ tiered-serial, $h$ and $h_1$ T-equivalent and $h \approx_V h_1$
  (by Lemma 5)
- $tser(h_1,n) = TRUE$ for some $n$, $Tequiv(h_1,n) = TRUE$, $Vequiv(h_1,h)$ = TRUE,
  for terms $h$, $h_1$ corresponding to $h$, $h_1$ respectively
  (by (3) and Lemma 6)
- $tserz(h) = TRUE$ (by (E34))

C. Example

We give an example of an algebraic proof of the tiered-serializability of a history $h_1$ which cannot be achieved by commuting non-conflicting steps. The proof derives a tiered-serial history $h$ from $h_1$ using equations (E1) – (E34) of IV.A. For $i=1,2,3$, let the transaction $T_i$ be given by:

$$\pi_1(i,x) \pi_2(i,y) \pi_3(i,z) r(2,i,y) r(2,i,z) w(2,i,x)$$

and the history $h_1$ of $\{T_1,T_2,T_3\}$ be $h_1 = r_{11}^r r_{12}^r r_{13}^r w_{11}^w w_{12}^w w_{13}^w r_{21}^r r_{22}^r r_{23}^r w_{22}^w w_{23}^w r_{32}^r w_{32}^w r_{33}^r w_{33}^w$

The notation for steps has been condensed, e.g. $r(j,i,x)$ becomes $r_{ji}^r$, and blocks of steps of interest are underlined. Now, by repeatedly commuting write steps to different data items ((E32), (E23) and (E24)), we have that $Vequiv(w_{12}^w w_{13}^w w_{21}^w w_{23}^w w_{31}^w w_{32}^w)$ is TRUE (9)

By commuting the write blocks comprising the single steps $w_{11}^w$ and $w_{13}^w$ ((E33), (E23) and (E24)) in the second term in (9), we have that $Vequiv(w_{12}^w w_{13}^w w_{21}^w w_{23}^w w_{31}^w w_{32}^w)$ is TRUE (10)

By commuting write steps ((E32), (E23) and (E24)), $Vequiv(w_{12}^w w_{13}^w w_{21}^w w_{23}^w w_{31}^w w_{32}^w)$ is TRUE (11)

By (9), (10) and (11) and transitivity of $Vequiv ((E22))$, $Vequiv(w_{12}^w w_{13}^w w_{21}^w w_{23}^w w_{31}^w w_{32}^w)$ is TRUE (12)

Thus, by (12), (E23) and (E24), $Vequiv(h_2,h_1) = TRUE$ (13)

where $h_2$ is the history $h_2 = r_{11}^r r_{12}^r r_{13}^r w_{11}^w w_{12}^w w_{13}^w r_{21}^r r_{22}^r r_{23}^r w_{22}^w w_{23}^w r_{32}^r w_{32}^w r_{33}^r w_{33}^w$

By repeatedly commuting non-conflicting read and write steps ((E31), (E30), (E23) and (E24)), $Vequiv(h,h_2) = TRUE$ (14)

where $h$ is the history $h = r_{11}^r w_{11}^w r_{12}^r w_{12}^w r_{13}^r w_{13}^w r_{21}^r r_{22}^r r_{23}^r w_{22}^w w_{23}^w r_{32}^r w_{32}^w r_{33}^r w_{33}^w$

By (13), (14) and transitivity of $Vequiv ((E22))$, $Vequiv(h,h_1)$ = TRUE (15)

By straightforward applications of (E6) - (E14), $h$ is clearly tiered-serial, i.e.

$tser(h,3)$ = TRUE (16)

By repeatedly commuting steps belonging to different transactions we have, by (E25) - (E27) and (E15) - (E19), that $Tequiv(h_1,h)$ = TRUE (17)

By (16), (17), (15) and (E34), we conclude that $tserz(h_1)$=TRUE as required.

V. Conclusions

We have given an algebraic theory of serializable schedules for a model of concurrent partitioned transactions. The main contribution of this work lies in its generality. Rather than presenting a neat algebraic method for a specific class of serializability problem, we have given a general algebraic method that is applicable to a very wide range of serializability problems. Practically, an algebraic approach may or may not be the best way of proving serializability in a given application. It is not the aim of this paper to determine if
an algebraic approach is suitable for a particular application.

The aim in this paper is to give the option of an algebraic approach if an alternative method of proof is required.

Most other work on algebraic specification of concurrency has been in the area of process algebras, notably CSP [7] and the \( \pi \)-calculus [15]. A comprehensive survey of different algebraic approaches to concurrency can be found in [1], although most approaches are applied to processes other than database transactions. An exception is the equational theory in [3] which is used to analyze the interplay between ACID properties of transactions. The initial-algebra approach in this paper has not been used in connection with database transactions. However, it is an established area of algebraic specification and standardized systems to support its use have been developed recently [2].

REFERENCES


