International Journal of Computer and Information Sciences, Vol. 14, No. 6, 1985

On Serializability[1]

J. A. Brzozowski[2] and S. Muro[3]

Received September 1984; revised October 1985

Concurrent execution of database transactions is desirable from the point of view of speed, but may introduce inconsistencies. A commonly used criterion of correctness of a concurrent execution of transactions is serializability, i.e., the equivalence of the execution to some serial schedule or schedules. In the literature several transaction models have been used and several different notions of serializability have been introduced. In this paper, we investigate the various serializability families in the general transaction model, in the two-step model, and in the restricted two-step model. We also examine these families in the multiversion database model.

KEY WORDS: Database; transaction; concurrency; serializability; families of schedules.

1. INTRODUCTION

Concurrency of database transactions is receiving a great deal of attention in the current literature.[1-7] This paper is concerned with some theoretical aspects of the problems encountered with concurrent execution of a set of transactions. In such a context, a simplified model of a database is usually used.[8-10] In fact, we assume that a *database* consists of a finite set

$$D = \{d_1, ..., d_{|D|}\}$$

[1] This research was supported in part by a Japan Society for the Promotion of Science Research Fellowship, in part by a Scientific Research Grant-in-Aid from the Ministry of Education, Science and Culture of Japan, and in part by the Natural Science and Engineering Research Council of Canada under Grant No. A1617. Most of this work was done when the first author was a Visiting Professor in the Department of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan.

[2] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

[3] Department of Applied Mathematics and Physics, Kyoto University, Kyoto 606, Japan.

of *data items*, where $|D|$ is the cardinality of D. Two types of accesses are permitted to each item $d \in D$, namely a *read access* $r(d)$ and a *write access* $w(d)$. At any given time each item can be accessed at most once, but several items may be read simultaneously or written simultaneously.

A transaction is usually considered to be a finite sequence of accesses (or actions),[8-10] such that, the execution of the sequence preserves the consistency of the database. Furthermore, it is normally assumed that the transactions are independent of each other in the sense that any serial execution of a given set of transactions is considered to be acceptable. Concurrent (interleaved) execution of a set of transactions is desirable to increase the speed, but may introduce inconsistencies.[8] The usual notion of correctness of a concurrent execution of a set of transactions is that of serializability,[8,9] i.e., the equivalence of the execution to some serial schedule.

One of the difficulties encountered in the literature is that various authors use different definitions of database, transaction, and equivalence of executions, and that frequently these notions are introduced only informally. In this paper, we use mathematically precise and explicit definitions of these terms. We use a more general notion of transaction,[11,12] namely one involving a partial order, and we define several notions of serializability. We then study the relationships among the various families of serializable executions. This is done for three types of transaction models: general,[3,11,12] two-step,[9] and restricted two-step.[9] We also examine these families in the multiversion database model.[1-3,11,13]

An early version of this work appeared as a technical report in Ref. 14. Further motivation and discussion of the definitions was given by Brzozowski.[12] We now summarize the relevant background material. Futher discussion of the literature is postponed until Section 6.

A *transaction* $T_i = (A_i, \leqslant_i)$ is a (finite) set A_i of accesses and an arbitrary partial order $\leqslant_i$ on A_i.[12] For $a, b \in A_i$, $a \leqslant_i b$ means that the access a must be performed before the access b. Let

$$\Sigma_i = \{r_i(X) \mid X \subseteq D\} \cup \{w_i(Y) \mid Y \subseteq D\}$$

be the alphabet of all the possible parallel accesses of a transaction T_i, where $r_i(X)(w_i(Y))$ represents the simultaneous reading (writing) of all the items in $X(Y)$. An *execution*[12] e of a transaction T_i is a word $e \in \Sigma_i^*$ such that all the accesses of A_i appear in e exactly once, no other accesses appear in e, and the partial order in which the accesses of A_i appear in e is consistent with $\leqslant_i$.

We are concerned with the execution of a finite set $A_J = \{T_1, ..., T_n\}$ of transactions. Formally, a *job* $J = (A_J, \leqslant_J)$ is a set A_J of transactions along with an arbitrary partial order $\leqslant_J$ on A_J.[12] In this paper however, we assume that $\leqslant_J$ is the trivial partial order containing only pairs of the type (T_i, T_i). Let J be a job, and let

$$\Sigma = \bigcup_{i=1}^{n} \Sigma_i$$

be the alphabet of all possible parallel accesses of J, where Σ_i is the set of accesses of transaction T_i, for $i = 1,..., n$. Also let E_i be the set of all executions of T_i. Then an execution of a job J is any word $w \in \Sigma^*$ which is in the "shuffle" of the E_i, as follows. The shuffle E_i \$ E_j of two sets E_i and E_j over disjoint alphabets Σ_i and Σ_j is the set of all the words of the form

$$s_1 t_1 \cdots s_m t_m \in (\Sigma_i \cup \Sigma_j)^*$$

where $m \geqslant 1$, $s_1,..., s_m \in \Sigma_i^*$, $t_1,..., t_m \in \Sigma_j^*$, $s_1 \cdots s_m \in E_i$, and $t_1 \cdots t_m \in E_j$. Since the shuffle operation in associative, the shuffle $\$_{i=1}^{n} E_i$ of n sets is uniquely defined by $E_1 \mathbin{\$} E_2 \mathbin{\$} \cdots \mathbin{\$} E_n$.

We assume that the database exists in some initial state where each item d has the value d_0. Sometimes it is convenient to introduce a fictitious transaction T_0 which has the single access $w_0(D)$ that writes all the initial database values. When an execution of a job J takes place, the database state may change as a result of write accesses. The final value of $d \in D$ that exists after an execution e will be denoted by $d(e)$. It is sometimes convenient to introduce a fictitious transaction T_f that has the single access $r_f(D)$ that reads all the database values. Clearly $r_f(d) = d(e)$. Let

$$\bar{D}(e) = (d_1(e),..., d_{|D|}(e))$$

be the final tuple of database values after execution e.

In general, the execution of a job also provides information to each transaction. We denote by $d(e)_i$ the value of d read by T_i during execution e. (By definition, such a read can take place at most once for each transaction.) The tuple $(d_1(e)_i,..., d_k(e)_i)$ of all the items read by T_i is denoted by $\tau_i(e)$. If T_i does not read d_j, then $d_j(e)_i$ is simply omitted from the tuple. Finally,

$$\tau(e) = (\tau_1(e),..., \tau_n(e))$$

is the tuple of values read by all the transactions during execution e.

We make the following assumption[12] concerning the writing of an item d by transaction T_i: $w_i(d)$ *depends* on all the values $r_i(d_j)$ which satisfy $r_i(d_j) \leqslant_i w_i(d)$, i.e. there exists some function $\Psi_{i,d}$, such that

$$w_i(d) = \Psi_{i,d}(r_i(d_1),..., r_i(d_k))$$

where $\{r_i(d_1),..., r_i(d_k)\} = \{r_i(d_j) \mid r_i(d_j) \leqslant_i w_i(d)\}$. Here $\Psi_{i,d}$ is treated as an uninterpreted function symbol.[9]

Let $J = (\{T_1,..., T_n\}, \leqslant_J)$ be a job and E the set of all executions of J. We define the following equivalence relations on E.[12] For all $i = 1,..., n$, and for all $e, e' \in E$,

$$e \sim_\delta e' \quad \text{iff} \quad D(e) = D(e')$$

$$e \sim_i e' \quad \text{iff} \quad \tau_i(e) = \tau_i(e')$$

$$e \sim_\tau e' \quad \text{iff} \quad e \sim_i e' \quad \text{for all} \quad i = 1,..., n$$

An execution e of $J = (\{T_1,..., T_n\}, \leqslant_J)$ is

(a) *δ-serializable* iff there exists a serial execution s of J such that $e \sim_\delta s$.

(b) *τ_*-serializable* iff there exist serial executions $s_1,..., s_n$ of J such that $e \sim_i s_i$ for all $i = 1,..., n$.

(c) *τ-serializable* iff there exists a serial execution s of J such that $e \sim_\tau s$.

(d) *piecewise serializable* iff there exist serial executions $s, s_1,..., s_n$ such that $e \sim_\delta s$ and $e \sim_i s_i$ for all $i = 1,..., n$.

(e) *serializable* iff there exists a serial execution s such that $e \sim_\delta s$ and $e \sim_\tau s$.

Let **D**, $\mathbf{T}_*$, **T**, **P**, and **R** denote the families of all executions that are δ-serializable, τ_*-serializable, τ-serializable, piecewise serializable, and serializable, respectively. The motivation for these definitions is as follows. The family **D** satisfies the database by providing the same final value for each item as would be obtained in some serial schedule. However, no attention is given to the transaction information. The families $\mathbf{T}_*$ and **T** do not care about the database but satisfy the transactions in some way. In $\mathbf{T}_*$ each transaction thinks that a serial execution took place, but each one may have a different execution in mind. In contrast to this, in **T**, all transactions think that one serial execution took place. In **P**, the database and each transaction think that a serial execution took place, but each may have a different one in mind. Finally, in **R** everybody sees one and the same equivalent serial execution.

Brzozowski[12] showed that each serializability family shown in Fig. 1 is nonempty. We denote by **E** the set of all executions and by **S** the set of all serial executions. The examples $e_0,..., e_6$ used in Ref. 12 were:

$$e_0 = r_1(a)\, r_2(b)\, w_2(a)\, w_1(b)\, r_3(a, b) \in \overline{\mathbf{D}} \cap \overline{\mathbf{T}}_*$$

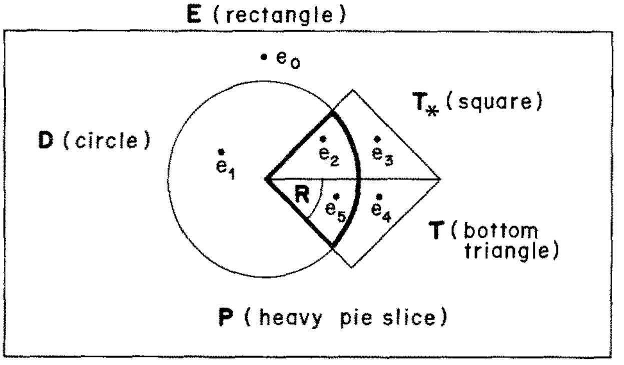

Fig. 1. Serializability families.

where $r_1(a) \leqslant_1 w_1(b)$ and $r_2(b) \leqslant_2 w_2(a)$;

$$e_1 = w_1(a)\, r_2(a)\, w_2(b)\, r_1(b) \in \mathbf{D} \cap \overline{\mathbf{T}}_*$$

where $w_1(a) \leqslant_1 r_1(b)$ and $r_2(a) \leqslant_2 w_2(b)$;

$$e_2 = r_2(a)\, w_1(a)\, r_1(b)\, w_2(b) \in \mathbf{D} \cap \mathbf{T}_* \cap \overline{\mathbf{T}}$$

where the transactions are the same as in e_1;

$$e_3 = r_1(a)\, r_2(a)\, w_1(a)\, w_2(a) \in \overline{\mathbf{D}} \cap \mathbf{T}_* \cap \overline{\mathbf{T}}$$

where $r_1(a) \leqslant_1 w_1(a)$ and $r_2(a) \leqslant_2 w_2(a)$;

$$e_4 = r_2(a)\, w_1(a)\, w_2(a) \in \overline{\mathbf{D}} \cap \mathbf{T}$$

where $r_2(a) \leqslant_2 w_2(a)$;

$$e_5 = r_2(b)\, w_1(a, b)\, w_2(a) \in \mathbf{D} \cap \mathbf{T} \cap \overline{\mathbf{R}}$$

where $r_2(b)$ and $w_2(a)$ are not related by $\leqslant_2$; and

$$e_6 = r_1(a)\, w_2(b)\, w_1(a) \in \mathbf{R} \cap \overline{\mathbf{S}}$$

where $r_1(a) \leqslant_1 w_1(a)$.

Given an execution e we define the "reads from" function ρ_e for e as follows:

$$\rho_e\colon \{r_i(d) \mid r_i(d) \in A_i,\ T_i \in A_J\} \rightarrow \{w_j(d) \mid w_j(d) \in A_j,\ T_j \in A_J\} \cup \{w_0(d)\}$$

where $\rho_e(r_i(d)) = w_j(d)$, if e has the form

$$e = u_1\, w_j(d)\, u_2\, r_i(d)\, u_3$$

where u_1, u_2, $u_3 \in \Sigma^*$ and u_2 does not have any d-writes, and $\rho_e(r_i(d)) = w_0(d)$ if

$$e = u_1 r_i(d)\, u_2$$

where u_1 has no d-writes.

One easily verifies that

$$e \sim_\tau e' \qquad \text{iff} \quad \rho_e = \rho_{e'}$$

2. THE TWO-STEP MODEL

In Papadimitriou's model[9] a transaction $T_i = (A_i, \leqslant_i)$ has one of the following three forms:

(a) $A_i = \{r_i(X), w_i(Y)\}$, $X, Y \subseteq D$, $r_i(X) \leqslant_i w_i(Y)$;

(b) $A_i = \{r_i(X)\}$, $X \subseteq D$—a "read-only" transaction;

(c) $A_i = \{w_i(X)\}$, $X \subseteq D$—a "write-only" transaction.

For this model the serializability families are the same as in the general model, as shown in the following.

Proposition 1. In the two-step model each serializability family of Fig. 1 is nonempty.

Proof. The following executions contain only two-step transactions, and one can verify that they appear in Fig. 1 as claimed.

(a) e_0, e_3, and e_4 are as in Section 1.

(b) $e'_1 = e_0 w_4(a, b) \in \mathbf{D} \cap \overline{\mathbf{T}}_*$.

(c) $e'_2 = r_1(a, b)\, r_2(a)\, w_2(a)\, r_3(a, b)\, w_1(b) \in \mathbf{D} \cap \mathbf{T}_* \cap \overline{\mathbf{T}}$.

To see that $e'_2 \notin \mathbf{T}$, note first that any serial execution beginning with T_3 will have $r_3(a)$ incorrect, i.e., different from that of e'_2. Any serial execution in which T_1 precedes T_3 will have $r_3(b)$ incorrect. This leaves $T_2 T_3 T_1$ as the only possibility. However, now $r_1(a)$ is incorrect. Thus $e'_2 \in \overline{\mathbf{T}}$. Since $e'_2 \sim_\delta T_1 T_2 T_3$, we have $e'_2 \in \mathbf{D}$. Finally $e'_2 \sim_1 T_1 T_2 T_3$, $e'_2 \sim_2 T_2 T_3 T_1$, and $e'_2 \sim_3 T_2 T_3 T_1$, showing that $e'_2 \in \mathbf{T}_*$.

(d) $e'_5 = r_1(a)\, w_1(b)\, r_2(b)\, w_3(b, d)\, r_4(d)$
$w_4(a, c, e)\, w_5(b, e)\, r_6(e)\, w_6(a, c, d)$, $w_2(c)\, w_7(a, b, d, e) \in$
$\mathbf{D} \cap \mathbf{T} \cap \overline{\mathbf{R}}$

This interesting example is due to *T*. Ibaraki.[4] Let the serial execution s be defined by

$$s = T_3 T_5 T_1 T_4 T_6 T_2 T_7$$

[4] Personal communication.

where the transactions are those of e'_5. Note that T_7 is a write-only transaction that writes the final values of a, b, d, and e in both e'_5 and s. Hence these two executions agree in these final values. Also, the last value of c is written by $w_2(c)$, which is determined by $r_2(b)$, which in turn reads $w_1(b)$ in both executions. Now $w_1(b)$ depends only on $r_1(a)$, which reads the initial value in both executions. Hence $e'_5 \sim_\delta s$.

Now consider the serial execution

$$t = T_1 T_2 T_3 T_4 T_5 T_6 T_7$$

One verifies that t and e'_5 have the same reads-from function. Hence $e'_5 \sim_\tau t$.

It remains to be shown that $e'_5 \notin \mathbf{R}$. For suppose there exists a serial execution u such that, $u \sim_\delta e'_5$ and $u \sim_\tau e'_5$. First, T_4 and T_6 must precede T_2 because all three write c, and the last value of c must be written by T_2. Second, T_1 must precede T_4 and T_6 because T_4 and T_6 write a and T_1 reads the initial value of a. Third, T_2 gets b from T_1, and T_3 and T_5 write b. Therefore neither T_3 not T_5 can occur between T_1 and T_2. If T_3 occurs after T_2, then T_4, which reads d from T_3, must also occur after T_2. This contradicts the first conclusion. Therefore T_3 must precede T_1 and, by a similar argument, T_5 must precede T_1. Altogether we must have the partial order shown in Fig. 2. Now, to obtain a total order from Fig. 2, we must either put T_4 between T_5 and T_6 or after T_6. In the first case $r_6(e)$ will read $w_4(e)$ which is wrong. If we put T_4 after T_6, then $r_4(d)$ reads $w_6(d)$ which is again wrong. Therefore the serial execution u cannot exist, and $e'_5 \notin \mathbf{R}$.

(e) Finally, note that $e'_6 = r_1(a)\, r_2(b)\, w_1(a)\, w_2(b)$ is in $\mathbf{R} \cap \overline{\mathbf{S}}$, and this concludes the proof. ▌

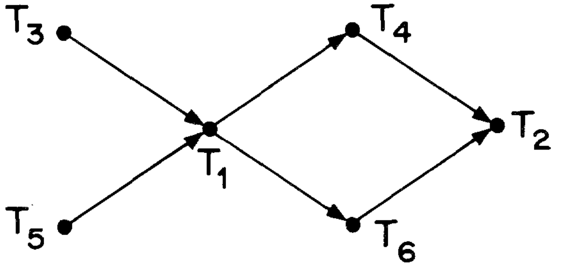

Fig. 2. Partial order obtained so far.

3. THE RESTRICTED TWO-STEP MODEL

A restricted version of the two-step model was also considered by Papadimitriou.[9] Here, a transaction is defined as in Section 2, with the additional restriction that if $A_i = \{r_i(X), w_i(Y)\}$ then $X \supseteq Y$. This means that every data item whose value is written by a transaction must first be read. This type of model was originally introduced by Stearns *et al.*[10] Although this restriction may seem rather minor at first glance, it is in fact a significant modification of the model.[9,10] The following proposition describes the serializability families for the two-step restricted model.

Proposition 2. In the two-step restricted model $\mathbf{T} = \mathbf{R}$, $\mathbf{D} = \mathbf{P}$, and the serializability classes are as shown in Fig. 3.

Proof. There are 7 serializability families in Fig. 1. We will show in Theorem 1 below that $\mathbf{T} = \mathbf{R}$, and in Theorem 2 that $\mathbf{D} = \mathbf{P}$. Assuming these results, there will be no examples like e_1, e_4, and e_5 here. We have

$$e'_0 = r_1(a, b)\, r_2(a, b)\, w_2(a)\, w_1(b)\, r_3(a, b) \in \overline{\mathbf{D}} \cap \overline{\mathbf{T}}_*$$

because neither $T_1 T_2$ not $T_2 T_1$ can produce the same final database values as e'_0, and r_3 reads these values. Also, e'_2, e_3, and e'_6 are in the restricted model. ▮

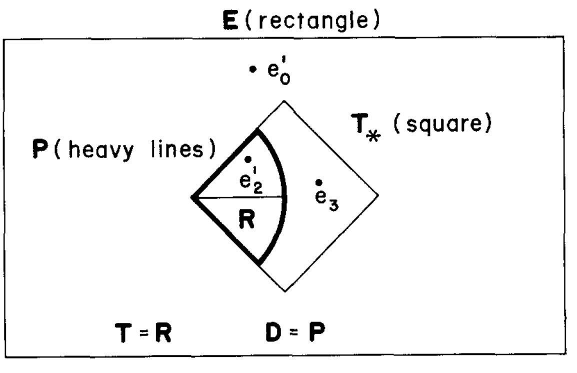

Fig. 3. Serializability families in the restricted two-step model.

Before proving Theorem 1 we need some preliminary results.

Lemma 1. Let s be a serial execution in which transaction T_i writes an item d before transaction T_j reads d. In other words, suppose s has the form:

$$s = u_1 w_i(Y)\, u_2 r_j(X)\, u_3$$

for some u_1, u_2, $u_3 \in \Sigma^*$, and $d \in X \cap Y$. Then $r_j(d)$ depends on $w_i(d)$, in the restricted two-step model.

Proof. Let all the write accesses in u_2 which write d be $w_1(Y_1),\dots, w_m(Y_m)$. In the restricted two-step model each $w_k(Y_k)$ must be preceded by $r_k(X_k)$ with $X_k \supseteq Y_k$, for $k=1,\dots, m$. Thus u_2 has the form:

$$u_2 = v_1 r_1(X_1)\, w_1(Y_1)\, v_2 r_2(X_2)\, w_2(Y_2) \cdots v_m r_m(X_m)\, w_m(Y_m)\, v_{m+1}$$

where $d \in X_k \cap Y_k$ for all $k=1,\dots, m$, and $v_1,\dots, v_{m+1}$ do not have any d-writes. Now if $m=0$, there are no d-writes in u_2 and $r_j(d)$ reads $w_i(d)$ in s. Otherwise, $r_j(d)$ reads $w_m(d)$ which depends on $r_m(d)$, which in turn reads $w_{m-1}(d)$, etc. Thus $r_j(d)$ depends on $w_i(d)$. ▮

Corollary 1. In the restricted two-step model, if e is δ-serializable and $w_i(d)$ appears in e then the final value $d(e)$ depends on $w_i(d)$.

Proof. Suppose $e \sim_\delta s$, where s is serial. Then also $er_f(D) \sim_\delta sr_f(D)$, and $d(s) = r_f(d)$ depends on $w_i(d)$ in s by Lemma 1. But $d(e) = d(s)$, and $d(e)$ depends on $w_i(d)$ in e. ▮

Corollary 2. In the restricted two-step model, if e is δ-serializable then $e \sim_\delta e'$ implies that each write access writes the same value in e as in e'.

Proof. Suppose $w_i(d)$ is a write access in e and e', and s is a serial execution such that $e \sim_\delta s$. By Lemma 1, $r_f(d) = d(s)$ depends on $w_i(d)$ in s, and hence in e and e'. If $w_i(d)$ writes a different value in e than in e', then $d(e) \neq d(e')$. This contradicts the fact that $e \sim_\delta e'$. ▮

Theorem 1. In the restricted two-step model $\mathbf{T} = \mathbf{R}$.

Proof. Suppose $e \in \mathbf{T}$, i.e., there exists a serial t such that $e \sim_\tau t$. If $e \not\sim_\delta t$, then there exists $d \in D$ such that $d(e) \neq d(t)$. Suppose first that d is last written by T_i in both e and t. In the two-step restricted model T_i must have $r_i(X)$ such that $d \in X$, and $w_i(d)$ depends on all $r_i(d')$, $d' \in X$. But, because $e \sim_\tau t$, all the reads in $r_i(X)$ must get the same values in both e and t. Therefore $d(e) = d(t)$. Consequently, if $d(e) \neq d(t)$, the last value of d must be written by some T_1 in t and by some $T_2 \neq T_1$ in e. Thus we can write

$$t = u_1\, r_2(X_2)\, w_2(Y_2)\, u_2 r_1(X_1)\, w_1(Y_1)\, u_3$$

for some u_1, u_2, $u_3 \in \Sigma^*$, where u_3 has no d-writes and $d \in X_1 \cap Y_1 \cap X_2 \cap Y_2$. Also, we must have

$$e = v_1 r_1(X_1)\, v_2 w_1(Y_1)\, v_3 w_2(Y_2)\, v_4$$

for some $v_1,\dots, v_4 \in \Sigma^*$, where v_4 has no d-writes. By Lemma 1, $r_1(d)$ depends on $w_2(d)$ in t. However, $r_1(d)$ cannot possibly depend on $w_2(d)$ in

e. Hence $e \not\sim_\tau t$, which is a contradiction. Therefore we must have $d(e) = d(t)$ for all $d \in D$, i.e., $e \sim_\delta t$. Altogether we have shown that $e \in \mathbf{R}$, and $\mathbf{T} \subseteq \mathbf{R}$. Since $\mathbf{R} \subseteq \mathbf{T}$, the theorem follows. ▌

Theorem 2. In the restricted two-step model $\mathbf{D} = \mathbf{P}$.

Proof. Suppose $e = u_1 r_i(X_i) u_2$ is δ-serializable. Let $\bar{u}_1$ be obtained from u_1 as follows. If $r_j(X_j)$ appears in u_1 but $w_j(Y_j)$ does not appear in u_1, remove $r_j(X_j)$ from u_1. After all such reads have been removed we have $\bar{u}_1$ remaining. Note now that, if we replace u_1 by $\bar{u}_1$ in e, $r_i(X_i)$ still gets the same information as it did in e.

We now claim that, if e is δ-serializable then the execution $\bar{u}_1$ obtained from any prefix u_1 of e is also δ-serializable. For let s be any serial execution δ-equivalent to e and let $\bar{s}$ be obtained from s by removing all the transactions which are not in $\bar{u}_1$. Then $\bar{u}_1 \sim_\delta \bar{s}$. To verify this suppose the final value of d written by $\bar{u}_1$ is different from that written by $\bar{s}$. If both values are written by the same transaction then, since $e \sim_\delta s$, the values must be the same by Corollary 2. If the final value of d is written by $w_1(X_1)$ in $\bar{u}_1$ and by $w_2(X_2)$ in $\bar{s}$, then these two writes appear in the order (w_1, w_2) in s and in the order (w_2, w_1) in e. But then $r_2(d)$, and hence $w_2(d)$, depends on $w_1(d)$ in s, but this is not true in e. This contradicts that $e \sim_\delta s$. Therefore we must have $\bar{u}_1 \sim_\delta \bar{s}$.

It now follows that any serial execution t beginning with $\bar{s} r_1(X_i)$ satisfies $e \sim_i t$. Thus $e \in \mathbf{T}_*$.

Altogether we have shown that $\mathbf{D} \subseteq \mathbf{T}_*$ which implies that $\mathbf{P} = \mathbf{D} \cap \mathbf{T}_* = \mathbf{D}$. ▌

The execution e_2' of Proposition 2 shows that $\mathbf{P} \neq \mathbf{R}$. However, under the additional restriction that e does not contain any read-only transactions, we show that $\mathbf{D} = \mathbf{R}$.

Theorem 3. In the restricted two-step model without read-only transactions $\mathbf{D} = \mathbf{R}$.

Proof. Suppose $e \sim_\delta s$ where s is serial. If $e \not\sim_\tau s$, then there must exist a read $r_i(X_i)$ and a variable $d \in X_i$ such that $r_i(d)$ gets a different value in e than it does in s. By assumption, T_i has a write access w_i which writes at least one item d'. By Lemma 1, the final value of d' depends on w_i and hence on $r_i(d)$. Since $e \sim_\delta s$, $r_i(d)$ must have the same value in both e and s. Thus we have a contradiction and the claim holds. ▌

Under the assumption that read-only transactions are not permitted, the restricted two-step model has only the families of Fig. 4. Examples which show that the families of Fig. 4 are not empty are:

$$e_0'' = e_0' w_3(a, b) \in \overline{\mathbf{T}}_*$$

$$e_3 \in \mathbf{T}_* \cap \overline{\mathbf{R}}$$

$$e_6' \in \mathbf{R} \cap \overline{\mathbf{S}}$$

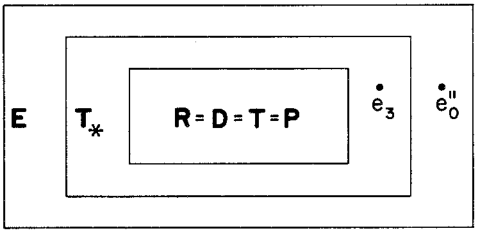

Fig. 4. Restricted model with no read-only transactions.

4. MULTIVERSION SERIALIZABILITY

The notion of serializability has been generalized in Refs. 1–3.11, 13, to the model in which the system keeps old copies of database values, called *versions*. In the limiting case an arbitrary number of versions may be kept, and this is the case we consider now.

Given a word $e \in \Sigma^*$ consider $T_0 e = w_0(D)\, e$, where T_0 is the fictitious initial write. In $T_0 e$ each item is written at least once. Thus, if $T_0 e = T_0 u_1 r_j(X_j)\, u_2$, and $d \in X_j$, we may assign to $r_j(d)$ any value $w_i(d)$ that occurs in $T_0 u_1$, including $w_0(d)$. To avoid confusion, we will now call a word $e \in \Sigma^*$ a *Σ-sequence* rather than an execution. Let R_e and W_e be the sets of read and write accesses in e. An *interpretation*[3] of e is a function

$$f\colon R_e \to W_e \cup \{w_0(d) \mid d \in D\}$$

which assigns to each $r_i(d) \in R_e$ any write $w_j(d)$ which precedes $r_i(d)$ in e. The interpretation in which the immediately preceding d-write is assigned to each d-read is called the *standard interpretation.*[3]

For notational convenience we will use the following convention. The interpretations will be called f_0, f_1, etc., and f_0 will always denote the standard interpretation. A Σ-sequence e and an interpretation f_i define an *f_i-execution* denoted by $e^{(i)}$. In this terminology $e^{(0)}$ is the normal, single-version, execution of e. For example, let e be the Σ-sequence:

$$e = w_1(a)\, r_2(a)\, w_2(b)\, r_1(a, b)$$

where $w_1(a) \leqslant_1 r_1(a)$, $w_1(a) \leqslant_1 r_1(b)$, and $r_2(a) \leqslant_2 w_2(b)$. There are eight interpretations of e as shown in Fig. 5. Each f_i-execution uniquely deter-

	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$r_1(a)$	a_1	a_0	a_0	a_0	a_1	a_1	a_1	a_1
$r_1(b)$	$w_2(b)$	b_0	b_0	$w_2(b)$	$w_2(b)$	b_0	b_0	$w_2(b)$
$r_2(a)$	a_1	a_0	a_1	a_0	a_1	a_0	a_1	a_0

Fig. 5. Interpretations of *e*.

mines the final database values and the transaction information. For instance,

$$D(e^{(6)}) = (a_1, \Psi_2(a_1))$$

$$\tau(e^{(6)}) = (a_1, b_0, a_1)$$

Consider now the serial Σ-sequence

$$s = w_1(a)\, r_1(a, b)\, r_2(a)\, w_2(b)$$

One verifies that $s^{(0)} \sim_\delta e^{(6)}$ and $s^{(0)} \sim_\tau e^{(6)}$. In this sense e is multiversion serializable; one can verify that it is not single-version serializable.

More formally, a Σ-sequence e is said to be *multiversion α-serializable* iff there exists an interpretation f_k such that $e^{(k)}$ is α-serializable, where α-serializable is any one of the following: δ-serializable, τ_*-serializable, τ-serializable, piecewise serializable, or serializable. If $\mathbf{F}$ is a single-version serializability family, we will denote by $\hat{\mathbf{F}}$ the corresponding multiversion serializability family.

Proposition 3. Multiversion serializability families are as shown in Fig. 6.

Proof. First note that $\mathbf{E} = \hat{\mathbf{T}}_*$. For let e be any Σ-sequence and let f_k be that interpretation which assigns to each read $r_i(d)$ the value $w_j(d)$ if

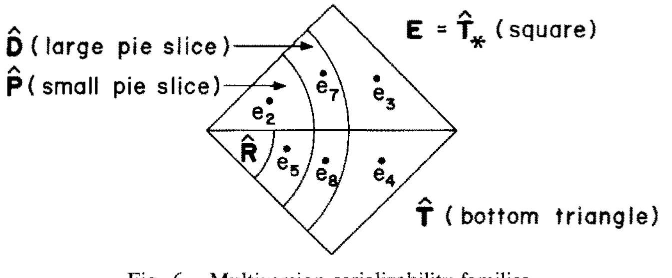

Fig. 6. Multiversion serializability families.

$w_i(d)$ precedes $r_i(d)$ in e, and it assigns the initial value of d if there is no $w_i(d)$ before $r_i(d)$ in e. Then the transaction information obtained by any transaction T_i from $e^{(k)}$ is the same as that obtained from any serial execution s_i beginning with T_i (ordered as in e). Thus

$$e^{(k)} \sim_i s_i^{(0)} \qquad \text{for all} \quad i = 1, \ldots, n$$

i.e., $e \in \hat{\mathbf{T}}_*$.

Now observe that all the reads in the Σ-sequences $e_2, \ldots, e_6$ of Section 1 can only be assigned the initial values. Hence each of these Σ-sequences has only the standard interpretation. One verifies that

$$e_3 \in \overline{\hat{\mathbf{D}}} \cap \overline{\hat{\mathbf{T}}}$$
$$e_4 \in \overline{\hat{\mathbf{D}}} \cap \hat{\mathbf{T}}$$

Next consider the families inside $\hat{\mathbf{D}}$. Although in the single version model $\mathbf{P} = \mathbf{D} \cap \mathbf{T}_*$, it is false that $\hat{\mathbf{P}} = \hat{\mathbf{D}} \cap \hat{\mathbf{T}}_*$. For let

$$e_7 = r_1(b)\, r_2(a)\, w_2(a, b)\, r_1(a)\, w_1(a)$$

where $r_1(a) \leqslant_1 w_1(a)$ but $r_1(b)$ is related neither to $r_1(a)$ not to $w_1(a)$, $r_2(a) \leqslant_2 w_2(a)$, and $r_2(a) \leqslant_2 w_2(b)$. Also let

$$s = r_2(a)\, w_2(a, b)\, r_1(a, b)\, w_1(a)$$

Then $e_7^{(0)} \sim_\delta s^{(0)}$, i.e. $e_7 \in \hat{\mathbf{D}}$. However, there does not exist a serial Σ-sequence s_1 such that $e_7^{(0)} \sim_1 s_1^{(0)}$, because no serial Σ-sequence can have T_1 reading both b_0 and $w_2(a)$ as $e_7^{(0)}$ does. Hence $e_7^{(0)} \notin \mathbf{P}$. The only other interpretation of e_7 is f_1 which assigns to $r_1(a)$ the initial value a_0. However, $e_7^{(1)}$ is not δ-serializable, so $e_7^{(1)} \notin \mathbf{P}$. Altogether, $e_7 \notin \hat{\mathbf{P}}$, and $\hat{\mathbf{P}}$ is a proper subset of $\hat{\mathbf{D}}$. One also verifies that $e_7 \notin \hat{\mathbf{T}}$, i.e., $e_7 \in \hat{\mathbf{D}} \cap \overline{\hat{\mathbf{P}}} \cap \overline{\hat{\mathbf{T}}}$.

To show that $\hat{\mathbf{D}} \cap \overline{\hat{\mathbf{P}}} \cap \hat{\mathbf{T}}$ is not empty, let

$$e_8 = r_1(b)\, w_2(a, b)\, r_1(a)\, w_1(a)$$

where $r_1(a) \leqslant w_1(a)$ but $r_1(b)$ is not related to $r_1(a)$ or $w_1(a)$. Let

$$s_1 = w_2(a, b)\, r_1(a, b)\, w_1(a)$$
$$s_2 = r_1(a, b)\, w_1(a)\, w_2(a, b)$$

Then $e_8^{(0)} \sim_\delta s_1^{(0)}$ and $e_8^{(1)} \sim_\tau s_2^{(0)}$, where f_1 is the sole nonstandard interpretation of e_8. Thus $e_8 \in \hat{\mathbf{D}} \cap \hat{\mathbf{T}}$. However, one verifies that $e_8 \notin \hat{\mathbf{P}}$.

Next consider subsets of $\hat{\mathbf{P}}$. One verifies that

$$e_2 \in \hat{\mathbf{P}} \cap \overline{\hat{\mathbf{T}}}$$
$$e_5 \in \hat{\mathbf{P}} \cap \hat{\mathbf{T}} \cap \overline{\hat{\mathbf{R}}}$$

where these are as in Section 1. Finally $e_6 \in \hat{\mathbf{R}} \cap \overline{\hat{\mathbf{S}}}$. ▮

5. THE TWO-STEP MULTIVERSION MODEL

The following proposition characterizes the serializability families in the two-step model.

Proposition 4. In the two-step model $\hat{\mathbf{D}} = \hat{\mathbf{P}}$ and the serializability families are as shown in Fig. 7.

Proof. The Σ-sequences e_3 and e_4 of Section 4 are two-step. In Theorem 4, we show that $\hat{\mathbf{D}} = \hat{\mathbf{P}}$. Now let

$$e_2'' = r_1(a, c)\, r_2(a)\, w_1(a)\, w_2(c)\, w_3(c)$$

Only the standard interpretation is possible here and one verifies that $e_2'' \in \hat{\mathbf{P}} \cap \overline{\hat{\mathbf{T}}}$. Also $e_6' \in \hat{\mathbf{R}} \cap \overline{\hat{\mathbf{S}}}$. The final example is a modified version of e_5', namely:

$$e = e_5'' = r_1(a)\, w_1(b)\, r_2(b)\, r_3(f)\, w_3(b, d)\, r_5(g)\, w_5(b, e)\, r_4(d)\, r_6(e)$$
$$w_4(a, c, e, f)\, w_6(a, c, d, g)\, w_2(a, c)\, w_7(a, b, e, f, g)$$

Let

$$s = T_3 T_5 T_1 T_4 T_6 T_2 T_7$$
$$t = T_1 T_2 T_3 T_4 T_5 T_6 T_7$$

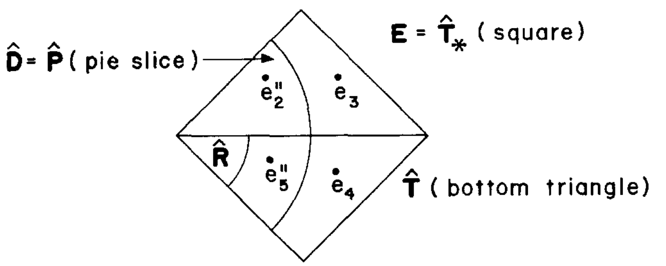

Fig. 7. Two-step multiversion families.

One verifies that $e^{(0)} \sim_\delta s^{(0)}$ and $e^{(0)} \sim_\tau t^{(0)}$, showing that $e \in \hat{\mathbf{D}} \cap \hat{\mathbf{T}}$. It remains to prove that no interpretation $e^{(k)}$ of e is serializable.

Note first that $r_1(a)$, $r_3(f)$, and $r_5(g)$ can only read the initial versions a_0, f_0, and g_0. We will show now that, for $e^{(k)}$ to be δ-serializable, f_k must be the standard interpretation. For suppose $r_2(b)$ reads b_0; then T_2 must precede T_1. But then $r_1(a)$ reads $w_2(a)$. Therefore $r_2(b)$ must read $w_1(b)$. Next suppose $r_4(d)$ reads d_0; then T_4 must precede T_3. Since T_3 reads f_0 and T_4 writes f we must have T_3 before T_4 which is a contradiction. Hence $r_4(d)$ can only be assigned $w_3(d)$. Similarly if $r_6(e)$ reads e_0 then T_6 precedes T_5. But then $r_5(g)$ cannot read g_0. Altogether we have shown that $e \in \hat{\mathbf{R}}$ implies $e^{(0)} \in \mathbf{R}$. However, the reader can easily verify that the arguments of Proposition 1 apply here also to show that $e^{(0)} \notin \mathbf{R}$. Therefore $e \notin \hat{\mathbf{R}}$. ▌

Theorem 4. In the two-step model $\hat{\mathbf{D}} = \hat{\mathbf{P}}$.

Proof. Suppose $e \in \hat{\mathbf{D}}$. Then there exists an interpretation f_1 and a serial Σ-sequence s such that $e^{(1)} \sim_\delta s^{(0)}$.

Any transaction T_i in e can be classified as δ-*live*[9] it it has a write $w_i(Y_i)$ and some final database value $r_f(a)$ depends on $w_i(b)$ for some $b \in Y_i$. Otherwise T_i in δ-*dead* and does not affect any final database values.

Suppose now $r_i(X_i)$ is a read in e. If T_i is δ-live then, for some a and b, $r_f(a)$ depends on $w_i(b)$ which in turn depends on all the $r_i(c)$, $c \in X_i$, in the two-step model. Since $e^{(1)} \sim_\delta s^{(0)}$ it follows that $e^{(1)} \sim_i s^{(0)}$.

On the other hand, if T_i is δ-dead construct an interpretation f_2 which is the same as f_1 on the reads of all the δ-live transactions and assigns the initial values to all the reads of all the δ-dead transactions. Then we have

$$e^{(2)} \sim_\delta s^{(0)}$$

$$e^{(2)} \sim_i s^{(0)} \qquad \text{if} \quad T_i \text{ is } \delta\text{–live}$$

$$e^{(2)} \sim_i s_i^{(0)} \qquad \text{if} \quad T_i \text{ is } \delta\text{–dead}$$

where s_i is any serial Σ-sequence beginning with T_i. Thus $e^{(2)} \in \mathbf{P}$ and $e \in \hat{\mathbf{P}}$. ▌

Our final proposition describes the serializability families in the restricted two-step multiversion model.

Proposition 5. In the restricted two-step model $\hat{\mathbf{R}} = \hat{\mathbf{T}} = \hat{\mathbf{D}} \neq \hat{\mathbf{T}}_* = \mathbf{E}$.

Proof. Suppose $e \in \hat{\mathbf{T}}$; then there exists an interpretation f_k such that $e^{(k)} \in \mathbf{T}$. By Theorem 1, $e^{(k)} \in \mathbf{R}$ and $e \in \hat{\mathbf{R}}$, giving $\hat{\mathbf{R}} = \hat{\mathbf{T}}$. Next suppose $e \in \hat{\mathbf{D}}$, i.e., $e^{(1)} \sim_\delta s^{(0)}$ for some f_1 and serial s. Let f_2 be the interpretation that agrees with f_1 on all the transactions that have a write, but assigns the initial values to all the read-only transactions. Then $e^{(2)} \sim_\delta e^{(1)}$. Now obtain t from s by removing all the read-only transactions from s and placing them all in front in any order. Then $e^{(2)} \sim_\delta t^{(0)}$ and $e^{(2)} \sim_\tau r^{(0)}$. Thus $e \in \hat{\mathbf{R}}$.

As before, example e_3 shows that $\hat{\mathbf{R}} \neq \hat{\mathbf{T}}_*$. ∎

6. CONCLUDING REMARKS

As we have mentioned before, some early papers on concurrency and serializability treat these notions rather informally. One of the objectives of this paper has been to stress the importance of mathematical precision; our results show that small changes in the definitions of transactions and serializability lead to different families of serializable schedules.

Secondly, we have used a more general model of transaction,[12] viewing it as a partially ordered set of tasks, and distinguishing between such a task specification and its execution.

Thirdly, we have shown that several notions of serializability can be used. Our definitions provide a framework for discussing the various models that have been used. For example, the paper by Papadimitriou[9] appears to be the first one where the concept of serializability used is what we have called δ-serializability. The work by Yannakakis[7] appears to be the first paper where an explicit distinction is made between "state serializability" (corresponding to our δ-serializability) and "view serializability" (corresponding to our serializability). Some interesting subclasses of serializable schedules and their characterizations are also given by Yannakakis.[7] Bernstein and Goodman[11] introduce the notion of what we have called τ-serializability. The concept of piecewise serializability has been introduced by Brzozowski.[12] For some interesting related work done recently the reader is referred to Papadimitriou and Yannakakis,[4] Tuzhilin and Spirakis,[5] and Vidyasankar.[6]

ACKNOWLEDGMENT

The authors wish to thank Prof. T. Ibaraki of Kyoto University for supplying the example for Proposition 1 and for other useful comments about this work.

REFERENCES

1. T. Hadzilacos and C. H. Papadimitriou, Algorithmic Aspects of Multiversion Concurrency Control, *Proc. ACM Symp. Principles of Database Systems*, pp. 96–104 (1985).

2. S. Muro, T. Kameda, and T. Minoura, Multiversion Concurrency Control Scheme for a Database System, *J. Comput. System Sci.* **29**:207–224 (1984).
3. C. H. Papadimitriou and P. C. Kanellakis, On Concurrency Control by Multiple Versions, *ACM Trans. Database Systems*, **9**:89–99 (1984).
4. C. H. Papadimitriou and M. Yannakakis, The Complexity of Reliable Concurrency Control, *Proc. ACM Symp. Principles of Database Systems*, pp. 230–234 (1985).
5. A. Tuzhilin and P. Spirakis, A Semantic Approach to Correctness of Concurrent Transaction Executions, *Proc. ACM Symp. Principles of Database Systems*, pp. 85–95 (1985).
6. K. Vidyasankar, Generalized Theory of Serializability, Department of Computer Science, Memorial University, St. John's, Newfoundland, Canada, Tech. Report No. 8510 (May 1985).
7. M. Yannakakis, Serializability by Locking, *J. ACM*, **31**:227–244 (1984).
8. K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger, The Notion of Consistency and Predicate Locks in a Database System, *Comm. ACM* **19**:624–633 (1976).
9. C. H. Papadimitriou, The Serializability of Concurrent Database Updates, *J. ACM*, **26**:631–653 (1979).
10. R. E. Stearns, P. M. Lewis II, and D. J. Rosenkrantz, Concurrency Control for Database Systems, *Proc. 17th Symp. Foundations of Computer Science*, pp. 19–32 (1976).
11. P. A. Bernstein and N. Goodman, Multiversion Concurrency Control-Theory and Algorithms, *ACM Trans. Database Systems*, **8**:465–483 (1983).
12. J. A. Brzozowski, On Models of Transactions, Department of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan, Tech. Report No. 84001 (April 1984).
13. T. Ibaraki and T. Kameda, Multiversion vs. Single-Version Serializability, Laboratory for Computer and Communication Research, Simon Fraser University, Burnaby, B. C., Canada, Tech. Report No. 83-1 (1983).
14. J. A. Brzozowski and S. Muro, On Serializability, Department of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan, Tech. Report No. 84012 (July 1984).

828/14/6-6