データベース・システム 54－5
（１９８６．７．２１）

多版モデルにおける並行処理制御のための先読みスケジューラ

加藤直樹 （神戸商科大学管理科学科） 茨木俊秀 （京都大学工学部数理工学科）
亀田恒彦 （サイモンフレーザー大学計算科学科）

これまで単版モデルにおけるデータベーストランザクションの並行処理制御のための先読みスケジューラーが提案されその特性が明らかになってきた。本報告では、それを多版モデルに拡張しその優れた性質について 述べる。単版モデルと同様に各トランザクションはあらかじめ読み取りあるいは書き込みを行うデータ項目を宣言しておく必要がある。先読みスケジューラーCSm(MC)はそれらの宣言された情報にもとずいて多版モデルにおける直列可能スケジュールのクラスMCに属するスケジュールを出力する。スケジューラーの中心部分の完成テストは、MC=MWW又はMWRWに対し多項式時間で実行できる。(単版モデルではWRWの完成テストは一般にNP完全。) そして宣言された操作の取消が起こっても正常に動作し，さらにMC=MWWにおいては、読み取り操作の追加が途中で生じても正常に動作する。これは単版モデルのスケジューラーにない優れた特徴である。またその並行処理能力は単版モデルのスケジューラーより高いことも示される。

Multiversion Cautious Schedulers for Database Concurrency Control

Naoki Katoh, Dept. of Management Science, Kobe Univ. of Commerce, Tarumi, Kobe, Japan
Ibaraki Toshihide, Dept. of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan
Tiko Kameda, Dept. of Computing Science, Simon Fraser University, Burnaby, B.C., Canada

This paper deals with multiversion cautious transaction schedulers for database concurrency control. As assumed in single-version cautious schedulers, each transaction, on arrival, predeclares its read and write sets. The multiversion cautious scheduler CSm(MC) reorders incoming requests from transactions in such a way that its output sequence belongs to MC, and it never resorts to rollbacks. We show that the completion test, which is the central part of the scheduler, can be done in polynomial time for MC=MWW and MWRW, while the WRW completion test is in general NP-complete in the single-version scheduler. It is also shown that multiversion cautious schedulers do not have cancellation anomalies, and that CSm(MWW) does not even have augmentation anomaly. Finally it is shown that in almost all cases, CSm(MC) allows more concurrency than the corresponding single-version scheduler, if concurrency is measured in terms of its fixed point set.

1. Introduction

A transaction scheduler in a database system must decide, for each arriving read/write request, if it can immediately be granted without violating serializability. In this paper we investigate multiversion cautious schedulers that never resort to rollbacks for the purpose of concurrency control.

In a single-version schedule, a read operation on a data item X reads the most recent value of X. In a multiversion schedule, on the other hand, a read operation can read any version of X. We distinguish a log (or history), h, which is merely a sequence of read and write operations, from a schedule, <h, I>, which is a log plus an interpretation I [PaK84]. An interpretation is a function that assigns to each read operation a version of the data item to be read. Let C and MC denote classes of logs that are serializable under the same serialization constraints, using the single-version and multiversion interpretations, respectively.

The purpose of this paper is to describe the multiversion cautious schedulers and to show several nice features of these schedulers, compared with the single-version counterparts already proposed in our recent papers [IKK85a, IKK85b, KIK86, KKI86]. See [IKK86] for the details and proofs of the results given in this paper.

In Section 2, we describe the database system model used in this paper. In Section 3, we present a general framework of the multiversion cautious scheduler, CSm(MC), where MC can be any subclass of MSR defined by serialization constraints. Section 4 discusses the MC-completion test, which is the most time consuming part of CSm(MC). We show that for MC=MWW and MWRW, MC-completion test can be done in polynomial time, while for other classes it remains NP-complete. In Section 5 we show that multiversion cautious schedulers do not exhibit cancellation anomaly that are observed in single-version cautious schedulers, and that some of these schedulers do not even exhibit augmentation anomaly. Finally,in Section 6, we propose version assignment rules that give priority to newer versions, and we show that for most classes, maintaining multiple versions of data items increases concurrency in the following sense. Namely, an input log h such that some steps of h are delayed by the single version cautious scheduler may encounter fewer or no delay at all during multiversion cautious scheduling. Throughout this paper, $\subset$ denotes a proper inclusion and, $\subseteq$ denotes an improper inclusion.

2. Database System Model

A database system consists of a set D of **data items** and a set $T = \{T_0, T_1, \ldots, T_f\}$ of **transactions**. A transaction consists of a totally ordered set of read and write steps, where a **read step(write) step** consists of a set of indivisible read (write) operations. The operations in a single step may be executed in any order. A **write operation**, $W_i[X]$, of transaction T_i creates a new **version** of data item X, and a **read operation**, $R_j[X]$, of transaction T_j returns the value of a version of X. Each data item is accessed by at most one read operation and at most one write operation of each transaction. If a transaction T_i both reads and writes a data item X, then $R_i[X]$ is ordered before $W_i[X]$ in T_i. The set of data items read (written) by a transaction is called its **read (write) set.** T_0 and T_f(consisting of a single write step $W_0[D]$ and read step $R_f[D]$ respectively) are fictious transactions called the **initial transaction**, and the **final transaction**, respectively. Any log starts with T_0 and ends with T_f.

Let Steps(T) denote the set of all read and write steps of a set T of transactions. A sequence over **Steps(T)** such that, for each i, its projection on Steps($\{T_i\}$) satisfies the total order for T_i is called a log. Given a log h over a set T of transactions, a **schedule** over T is a pair s = <h, I>, where I is a mapping, called an **interpretation** [PaK84] or **version assignment**, from the set of read operations into the write operations such that $I(R_j[X])$ must precede $R_j[X]$. The **standard interpretation** [PaK84] I^* associated with h is defined by $I^*(R_j[X]) = L_h(R_j[X])$, where $L_h(R_j[X])$ denotes the last write operation on X that precedes $R_j[X]$ in h. Clearly, any single-version schedule can be interpreted

as a multiversion schedule with the standard interpretation.

In order to represent a schedule s = <h, I> we often use a visual representation, called the linear representation, in which the steps in h are arranged linearly from left to right. Each write operation $W_i[X]$ is changed to $W_i[x_i]$ and each read operation $R_j[X]$ is changed to $R_j[x_i]$ if $I(R_j[X]) = W_i[X]$.

Let h and h' be two logs over T. Two schedules s=<h, I> and s'=<h', I'> are said to be equivalent, written $s \equiv s'$, if I=I'. A log that does not interleave steps from different transactions is called serial. A serial log with the standard interpretation is called a serial schedule. A multiversion schedule s is said to be serializable, if thre exists a serial schedule s' such that $s' \equiv s$. A log h is called serializable if there exists an interpretation I such that the schedule <h, I> is serializable, and such an I is called a serializing interpretation.

The transaction IO graph or TIO graph [IKM82, IKM83] for s, denoted by TIO(s), is a labeled directed multigraph with the node set $\mathbf{T} \cup \mathbf{T'}$ and the arc set A, where T' is given below. If T_j reads X from T_i, it has a reads-from arc $(T_i, T_j) \in A$ labeled by X (denoted by $(T_i, T_j):X$). If T_i performs $W_i[Y]$ but no other transaction reads Y from T_i, then $W_i[Y]$ is said to be a useless write [Set82], and we introduce a dummy node $T'_i \in \mathbf{T'}$ together with a dummy arc $(T_i, T'_i):Y$. Note that at most one dummy node is introduced for each transaction. Dummy nodes will be represented by small circles without labels.

Let s = <h, I> be a schedule. A total order $<<$ on the set of nodes of TIO(s) is a disjoint-interval topological sort or DITS [IKM82, IKM83], if it satisfies the following two conditions for any T_i, T_j, T_h, and T_k in $\mathbf{T} \cup \mathbf{T'}$:

(a) if $T_i << T_j$ then there exists no path from T_j to T_i in TIO(s), and

(b) [Exclusion rule[Set81]] for any two arcs labeled by X in TIO(s), $(T_h, T_i):X$ and $(T_j, T_k):X$, such that $h \neq j$, either $T_i << T_j$ or $T_k << T_h$ holds.

If $T_i << T_j$ in a DITS, we say that T_i is serialized before T_j.

Theorem 2.1 [IKM82]. A schedule s is serializable if and only if TIO(s) has a DITS which orders T_0 first and T_f last. □

Based on serialization constraints, called write-write(ww), write-read(wr), read-write(rw), and read-read(rr), classes of logs, MWW, MWR, MRW, MRR, etc., have been introduced [IbK83]. A log h belongs to a certain class MC if there exists an interpretation I for h such that <h, I> is equivalent to a serial schedule satisfying the set c (ww, wr, etc.) of imposed constraints. The single-version counterparts of these classes are called, WW, WR, RW, etc [IKM82]. Because of its importance in this paper, the union of all wr- and rw-constraints will be denoted by an abbreviation wrw. Similarly, we use MWRW (WRW) to denote the class of multiversion (single-version) log serializable under the wrw-constraints. In general, for a set of constraints c, let MC (C) stand for the class of multiversion (single-version) logs serializable under the constraints in c. Any one of the above types of constraints can be indicated in TIO(s) by constraint arcs called, ww-arcs, wr-arcs, etc. For a given set c of constraints, let $TIO_c(s)$ denote the TIO graph for s augmented by all the constraint arcs corresponding to c.

Theorem 2.2 [IKM82, IKM83]. A schedule s is serializable under a set c of constraints if and only if $TIO_c(s)$ has a DITS which orders T_0 first and T_f last. □

3. Cautious Scheduler

Let <P, I> be a partial schedule, where P is the output log that has so far been generated and I is its interpretation, and let q denote the current request, i.e., the step of operations being examined for granting or delaying. We introduce a list DEL of delayed steps, and a set PEND = $\cup$ Steps(T_i) - {the steps in Pq} of pending steps, where the union is over all $T_i \in \mathbf{T}$ such that the first step of T_i has already arrived. We note that PEND consists of two kinds of steps: those which are delayed and those which have predeclared but have not arrived yet. It is assumed that each transaction issues its

next request only if it has no delayed request.

Given <P, I>, q, and PEND, the multiversion cautious scheduler CSm(MC) must perform the **MC-completion test**, i.e., it must determine if it is possible to complete the partial schedule <P, I> by appending to it a sequence $qQR_f[D]$ and an interpretation I' such that (i) Q is a sequence over PEND, (ii) the order of steps in Q is consistent with that among the steps of each transaction, and (iii) the resulting log $PqQR_f[D]$ belongs to the given class MC, with a serializing interpretation II' for $PqQR_f[D]$, where II' is the union of interpretations I and I'.

In response to each new request q, CSm(MC) performs the MC-completion test. If the test fails, then it delays q and appends it to DEL. If the completion test succeeds, on the other hand, step q is granted, and the steps in DEL are reexamined one after another in order to see if they can be granted. The following is the precise description of the scheduling algorithm.

Procedure CSm(MC)

CS0: [Initialization] Let P := $W_0[D]$, q := the first request (issed by the first arriving transaction T_1), I := the empty function (P has no read operation yet), DEL := Λ (the null queue), and PEND := Steps(T_1) - {q}.

CS1: [Test the current request q] Apply the MC-completion test to the partial schedule <P, I> and q. If it fails, then go to CS2. Otherwise, grant q and go to CS3.

CS2: [q was delayed] Do one of the following, depending on the current request q.
(a) q $\in$ DEL and all the other steps, if any, in DEL are marked: Erase all marks from the steps in DEL and go to CS5.

CS3: [q was granted] Let P:= Pq, and if q is a read step, define I(p) fro each read operation in q as given by the MC-completion test of Step CS1. Unmark all the marked steps, if any, in DEL, and if q $\in$ DEL, delete q from DEL.

CS4: [Pick the next q in DEL] If DEL $\neq \Lambda$, then let q be the first step in DEL. Return to CS1.

CS5: [Wait for the next arriving request] Let q be the next arriving request. If q is the first step of a new transaction T_j, then let PEND := PEND $\cup$ Steps(T_j) - {q}. Return to CS1. □

Theorem 3.1. [IKK85a] For MC = MSR, MWW, MWR, MRW, MRR, and MWRW, and given an input sequence h' of steps generated by a set **T** of transactions, CSm(MC) always produces a log h $\in$ MC over T such that the order among the steps of each $T_i \in$ **T** is preserved in h. □

4. Completion Test

Though the MC-completion test can in general be quite time-consuming, Theorem 4.3 below will show that it can be done in polynomial time for MC = MWW and MWRW.

Definition 4.1 The **active TIO graph** (or **ATIO graph**, for short), denoted by ATIO(<P, I>; q; PEND), has a node set that consists of the transactions whose steps are in Pq and/or PEND, the final transaction T_f, and some **dummy nodes**, which are the terminal nodes of the dummy arcs below. Its arc set consists of two disjoint subsets, A and A'. Subset A contains those reads-from arcs and dummy arcs representing <P, I>. Each write operation $W_i[X]$ of a step in {q} $\cup$PEND is represented by a dummy arc (T_i, T'_i):X, called a **pending write arc.** □

In what follows we draw the arcs in A thick and those in A' thin. In addition, if q is a write step, the corresponding dummy arcs will be drawn thick. Thus we are pretending as if it had been granted. The dummy nodes are indicated by small circles. Also, as a reminder, we indicate a not-yet-granted read operation on X by a transaction T_i as a dangling arc to node T_i labeled by X.

Example 4.1 Consider the following partial schedule and pending steps.

$$<P, I> = W_0[x_0, y_0]R_1[x_0]R_2[y_0], \quad q = W_2[X], \quad PEND = \{W_1[X]\}.$$

The active TIO graph, ATIO(<P, I>;q; PEND) is shown in Fig. 4.1(a). □

DITS for ATIO graphs plays a central role in our MC-completion test. To take serialization constraints into account, we add constraint arcs to ATIO(<P, I>;q; PEND). Let A and B be two steps in Pq ∪ PEND. We write A < B if either A precedes B in Pq, or A ∈ Pq and B ∈ PEND. For a given set c of constraints, if a step A of T_i and a step B of T_j give rise to a c-constraint and A < B, then a constraint arc (T_i, T_j), called a c-arc, is introduced. For a set c of constraints, let $ATIO_c$(<P, I>;q; PEND) stand for the active TIO graph augmented by the c-arcs. In example 4.1, $ATIO_{wrw}$(<P, I>;q; PEND), shown in Fig. 4.1(b), has an rw-arc (T_1, T_2), since $R_1[X]$ precedes $W_2[X]$ in Pq. As shown below in Theorem 4.1, testing whether a given TIO or ATIO graph has a DITS can sometimes be facilitated by the "exclusion closure" defined as follows [IKM82, IKM83]. Let (T_h, T_i):X and (T_j, T_k):X be two arcs in $TIO_c(s)$ or $ATIO_c$(s;q; PEND), where h ≠ j. If there is a path in the graph from T_h to T_k, we introduce an unlabeled exclusion arc (T_i, T_j). Their **exclusion closures**, denoted by $TIO_c^*(s)$ and $ATIO_c^*$(s;q; PEND) are obtained from $TIO_c(s)$ or $ATIO_c$(s;q; PEND), respectively, by adding arcs (T_0, T_i) and (T_i, T_f) for all $T_i \neq T_0$, T_f, and all exclusion arcs. It can be shown that the exclusion closure is unique and does not depend on the order of adding the exclusion arcs.

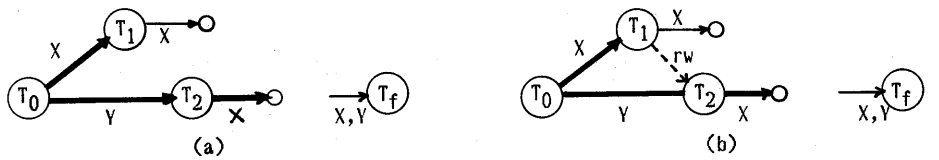

Fig. 4.1. Active TIO graphs for Example 4.1. (a) ATIO(<P, I>;q; PEND); (b) $ATIO_{wrw}$(<P, I>;q; PEND).

Theorem 4.1. [IKM82, IKM83]
(a) $TIO_{ww}(s)$ ($ATIO_{ww}$(s;q;PEND)) has a DITS if and only if $TIO_{ww}^*(s)$ ($ATIO_{ww}^*$(s;q;PEND)) is acyclic.
(b) $TIO_{wrw}(s)$ ($ATIO_{wrw}$(s;q;PEND)) has a DITS if and only if $TIO_{wrw}^*(s)$ ($ATIO_{wrw}^*$(s;q;PEND)) is acyclic.

The MC-completion test is performed based on G'=$ATIO_c$(<P, I>;q; PEND) with a partial schedule <P, I>, the current request q issued by T_j, and a set of pending steps PEND. Note that even if G' has a DITS, the DITS may assign the version to be written by a pending write operation to the current read request. For MC=MWRW, however, this does not happen, since rw-arc from each read operation $R_j[X]$ in q to all pending write operation on X forces T_j to be ordered before any transaction with a pending write on X. For MC=MWW, this is prevented by introducing an rw-arc from T_j to each T_k with a pending write operation on X ∈ S for $q=R_j[S]$. In the subsequent discussion, therefore, we always assume that these rw-arcs are included in $ATIO_{ww}$(<P, I>;q; PEND). Combining this with Theorem 4.1, we immediately obtain the following theorem.

Theorem 4.2. The MC-completion test for class MC=MWW or MWRW can be done in polynomial time. □

On the other hand, for the other classes, the MC-completion test is NP-complete, as shown below.

Theorem 4.3. [IKK86] For MC = MSR, MRW, or MWR, the MC-completion test is NP-complete. □

In view of Theorem 4.3, multiversion cautious schedulers CSm(MC) that has practical importance appear to be limited to CSm(WW) and CSm(WRW). In the rest of this paper, we investigate various aspects of multiversion cautious schedulers, placing particular emphasis on these two schedulers.

5. Cancellation and Augmentation Anomalies

As stated earlier, in our model, each transaction upon arrival predeclares its read set and write set. In real situation, however, it will be more convenient if transactions can cancel some of their predeclared operations. Some of the single-version cautious schedulers may block when predeclared operations are canceled, namely, they exhibit **cancellation anomaly** [IKK85a,b]. The following theorem shows that multiversion cautious schedulers CSm(MC) do not share such undesirable feature.

Theorem 5.1. [IKK86] CSm(MC) does not exhibit cancellation anomaly for any class MC introduced in Section 2. □

Next, we consider the opposite situation, in which transactions want to expand their predeclared read and/or write set. It is easy to see, however, that for any CSm(MC) of interest, the addition of a new write step may cause scheduler blocking. Therefore, we consider only the addition of new read steps, and say that a scheduler exhibits **augmentation anomaly**, if the addition of some read steps, not predeclared, can cause scheduler blocking. All single-version cautious schedulers studied in previous papers [IKK85a, KIK85, KKI86] exhibit augmentation anomaly as easily checked. Surprisingly, however, some multiversion cautious schedulers do not exhibit augmentation anomaly.

Theorem 5.2.[IKK86] (i) For classes MC=MSR, MWW, MRW, CSm(MC) does not exhibit augmentation anomaly. (ii) For other classes MC, however, CSm(MC) exhibits augmentation anomaly. □

We give here an example of augmentation anomaly for MC=MWR.

Example 5.1. Consider $\langle P, I\rangle = W_0[D]R_1[x_0]W_1[x_1]R_2[x_1]W_2[y_2]$, $q=R_3[Y]$, $PEND=\{W_1[Z], W_3[Y]\}$. $ATIO_{wr}(\langle P, I\rangle; q; PEND)$ is shown in Fig. 5.1(a). It is easy to see that this graph has a DITS. If T_1 declares a new read step $R_1[Y]$, a new pending read arc and a wr-arc are added and the graph shown in Fig. 5.1(b) is obtained, in which a cycle is detected. Since the cycle is due to three already granted operations $W_1[x_1]$,$R_2[x_1]$ and $W_2[y_2]$, it does not vanishes even if the current request $q=R_3[Y]$ is delayed. Thus CSm(WR) blocks. □

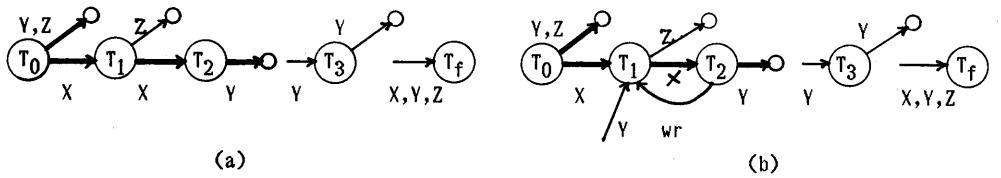

Fig. 5.1. Augmentation anomaly. (a) $ATIO_{wr}(\langle P, I\rangle; q; PEND)$; (b) Addition of $R_1[Y]$.

From these results, it is seen that CSm(MWW) is particularly important in practice, because it runs in polynomial time and has no cancellation or augmentation anomaly. The absence of augmentation anomaly in CSm(MWW) implies that it is required for a transaction to predeclare only its write set.

6. Version Assignment Rules and Fixed Point Sets of Cautious Schedulers

In this section, we investigate the version assignment rules in the completion test. Intuitively, we feel that most recent version that is feasible should be assigned. Therefore, we shall adopt the following rule.

Rule 0: Assign the most recent version to each read operation in q, if it results in a serializing

interpretation.

This rule does not specify what should be done if it fails. Thus, we will introduce two more rules to deal specifically with CSm(MWW) and CSm(MWRW). According to Theorems 4.1 and 4.2, the completion test for these schedulers can be done by the acyclicity test of $G^{\sharp}=ATIO_c^{\sharp}(<P, I>;q; PEND)$ with c=ww or wrw, which can be accomplished by successive elimination of source nodes as follows, starting with $G^{\sharp}$.
(1) Halt if $G^{\sharp}$ has no source node; the original ATIO graph is not acyclic.
(2) Choose a source node and eliminate it from $G^{\sharp}$ together with the arcs incident to it. Rename the resulting graph as $G^{\sharp}$. Halt if $G^{\sharp}$ has no node; the original ATIO graph is not acyclic. Otherwise return to (1). □

Theorefore, the method of version assignment depends on the order of source node elimination. The second rule is stated in this context.

Rule 1: Suppose that $q = R_j[S]$ and T_j has become a source node. If thre are source nodes other than T_j, eliminate all other source nodes first before eliminating T_j.

In order to state the rule for the MWRW-completion test, we assume that the data items in D are totally ordered. We use the term "smaller" to represent this total order.

Rule 2: Suppose that $q = R_j[S]$ and T_j has a source node. If there are source nodes other than T_j, eliminate all other source nodes in the following order before eliminating T_j: for transactions T_i and T_k with write operations $W_i[Y]$ and $W_k[Z]$, respectively, eliminate T_i before T_k if Y is "smaller" than Z, and break ties, by eliminating older version first.

Theorem 6.1. [IKK86] (i) Both Rules 1 and 2 are compatible with Rule 0, and will generate the standard interpretation if it is possible.
(ii) If Rule 1 is used in the MWW-completion test, then for each $X \in S$ the version of X that is assigned to $R_j[X]$ is the most recent possible. □

Let h be a log over a set T of transactions. Let $MC^{\sharp}$ denote the **fixed point set** [KuP79, Pap82], i.e., the set of logs h granted without delays, that is to say, no step of h is ever put in DEL by CSm(MC). The fixed point set gives a measure of concurrency allowed by a scheduler. The set $MC^{\sharp}$ clearly depends on the interpretation specified by CSm(MC) during the MC-comletion test. We adopt the version assignment Rule 0, and where applicable, Rule 1 (in CSm(MWW)) and Rule 2 (in CSm(MWRW)). We have the following characterizations of $MC^{\sharp}$.

Theorem 6.2. [IKK86] (i) For any set c of constraints introduced in Section 2, we have $C \subseteq MC^{\sharp} \subseteq MC$, where C is the class of logs that are serializable under constraints c in a single-version system.
(ii) For any constraint set c, not containing both ww- and wr-constraints, we have $C \subset MC^{\sharp} \subset MC$.
(iii) For a constraint set c that contains both ww- and wr-constraints, we have $C = MC^{\sharp} = MC$.
(iv) For two-step transactions, we have $WW = MWW^{\sharp} \subset MWW$. □

Fig. 6.1 shows the inclusion relationships among MC and $MC^{\sharp}$, where MC=MWW and MWRW.

References

[BeG83] P. A. Bernstein and N. Goodman, Multiversion concurrency control-theory and algorithms, ACM Trans. Database Systems 8, 4(1983), 465-483.

Dept. of CS, Simon Fraser Univ., Dec. 1982.
[IbK83] T. Ibaraki and T. Kameda, Multiversion vs. single-version serializability, LCCR TR83-1, Lab. for Computer and Commnunications Res., Simon Fraser Univ., 1983.
[IKM83] T. Ibaraki, T. Kameda and T. Minoura, Disjoint-interval topological sort: a useful concept in serializability theory, Proc. 9th Int. Conf. on VLDB, Oct/Nov. 1983, 89-91.
[IKK85a] T. Ibaraki, T. Kameda and N. Katoh, Cautious transaction schedulers for database concurrency control, LCCR Tech. Rep. 85-6, Lab. for Computer & Communications Res., Simon Fraser Univ., 1985.
[IKK85b] 茨木俊秀，亀田恒彦，加藤直樹，Cautious Scheduler（先読みスケジューラー）による並行処理制御，情報処理学会データベースシステム研究会資料，１９８５年７月．
[IKK86] T. Ibaraki, T. Kameda and N. Katoh, Multiversion cautious schedulers for database concurrency control, LCCR Tech. Rep. 86-2, Lab. for Computer & Communications Res., Simon Fraser Univ., 1986.
[KIK85] N. Katoh, T. Ibaraki and T. Kameda, Cautious transaction schedulers with admission control, ACM Trans. Database Systems 10, 2 (1985), 205-229.
[KKI86] N. Katoh, T. Kameda and T. Ibaraki, A cautious scheduler for multi-step transactions, To appear in Algorithmica 1 (1986).
[KuP79] H. T. Kung and C. H. Papadimitriou, An optimality theory of concurrency control for databases, Proc. ACM-SIGMOD Int. Conf. on Management of Data, Boston, May 1979, 116-126.
[Lau81] G. Lausen, Serializability problems of interleaved transactions in Lecture Notes in Computer Science 123; Trends in Information Processing, Springer Verlag, Berlin, 1981.
[Pap82] C. H. Papadimitriou, A theorem in database concurrency control, J. ACM 29, 4 (1982). 998-1006.
[PaK84] C. H. Papadimitriou and P. C. Kannelakis, On concurrency control by multiple versions, ACM Trans. Database Systems 9, 1 (1984), 89-99.
[Set81] R. Sethi, A model of concurrent database transactions, Proc. 22nd IEEE Symp. Foundation of Comp. Sci., Oct. 1981, 175-184.
[Set82] R. Sethi, Useless actions make a difference: Strict serializability of database updates, J. ACM 29, 2 (1982), 394-403.

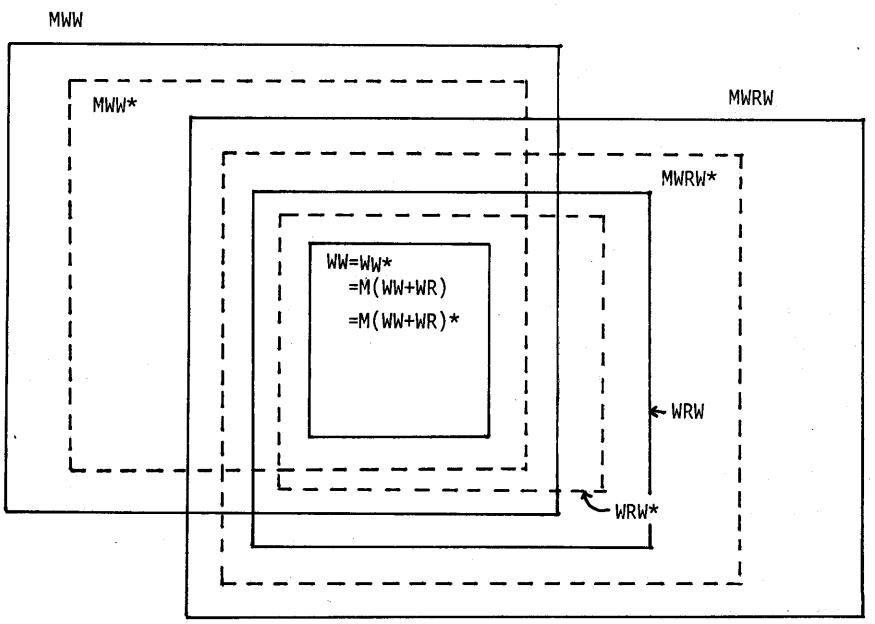

Fig. 6.1 Inclusion relationships among MC and MC* with MC=MWW and MWRW.