Disjoint-Interval Topological Sort: A Useful Concept in Serializability Theory (Extended Abstract)

T. Ibaraki+, T. Kameda++ and T. Minoura+++

+ Dept. of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Japan 440
++ Dept. of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
+++Dept. of Computer Science, Oregon State University, Corvallis, Oregon, U.S.A. 97331-4602

1. Introduction

The theory of serializability for concurrency control of databases has been extensively studied [ESWA-76, STEA-76, BERN-79, PAPA-79, SETH-81]. In this paper, we introduce a unifying concept in the theory, called disjoint-interval topological sort (DITS, for short), and discuss its applications, including a number of new results.

We prove that the existence of a DITS for the transaction IO graph (Section 3) associated with a schdule is a necessary and sufficient condition for serializability. The notion of DITS captures the essence of serializability and most known results on the characterization of serializable schedules follow directly from this main theorem.

The most important contributions of the DITS are its appeal to intuition and its wide applicability. It is not only useful as an analysis tool, as we demonstrate in this paper, but it also provides a useful aid to a scheduler [KATO-83]. The concept of DITS can be easily extended to the multi-version case [STEA-76, REED-79, MURO-81, BERN-82, PAPA-82, IBAR-83].

We demonstrate a class of schedules, called WR+RW (see Section 5), which is the largest class of serializable schedules currently known that is polynomially recognizable. We also state some NP-completeness results.

2. Database System Model and Schedule

A database system consists of a set D of data items and a set T = {To, T1, T2, ..., Tn, Tf} of transactions. The steps of a transaction are a partially ordered set of read and write operations [BERN-82]. A read operation Ri[X] of transaction Ti returns a value of data item X, and write operation Wi[X] creates a new value for X. To and Tf are fictitious transactions, called the initial transaction and final transaction [PAPA-79], respectively. To is a write-only transaction which "writes" the initial value of each data item, and Tf is a read-only transaction which "reads" the final value of each data item after all other transactions have completed. Each data item is accessed by at most one read and at most one write operation of each transaction.

Let OP(T) denote the set of all read and write operations of a set T of transactions. A schedule [ESWA-76] s over T is a pair (OP(T), <s), where <s is a total order on OP(T) consistent with the partial order among the operations of each transaction.

In order to represent the total order <s, we simply write operations from left to right in the order of <s (see [PAPA-79]).

If Wj[X] is the last write operation on X preceding Ri[X] in a schedule s, we say that Ti reads X from Tj in s. Two schedules s and s' are equivalent, written s $\equiv$ s', if for each X, i and j, Ti reads X from Tj in s iff Ti reads X from Tj in s'.

A schedule s is said to be serializable if there is a serial schedule s' such that s $\equiv$ s' [ESWA-76, BERN-79, PAPA-79]. SR denotes the set of all serializable schedules.

3. Transaction Input/Output Graph

Definition 3.1. Let s = (OP(T), <s) be a schedule over a set T of transactions. The transaction IO graph, denoted by TIO(s), is a labeled multigraph with the node set T$\cup$T' and the arc set A. If Tj reads X from Ti, there is an arc (Ti, Tj) $\in$ A labeled by X (denoted by (Ti, Tj):X). If Ti writes on a data item Y and if no other transaction reads it from Ti, then we introduce a distinct dummy node T'i $\in$ T' together with a dummy arc (Ti, T'i):Y. There are no other nodes or arcs in TIO(s). ▯

The version graph [STEA-76] and transaction dag [SETH-81] are similar to the TIO graph defined above, except for the dummy nodes and arcs. The TIO graph for the following schedule is shown in Fig. 1.

b = Wo[X]W1[X]R2[X]W3[X]W2[X]R4[X]W5[X]Rf[X].

4. Disjoint-Interval Topological Sort

We define an interval to be a set of all arcs that have the same label and originate from the same node.

Definition 4.1. Let (Th, Ti):X and (Tj, Tk):X be any two arcs labeled by X in TIO(s), where h ≠ j. A total order << on the set of nodes of TIO(s) is a disjoint-interval topological sort (DITS, for short), if it satisfies the following three conditions:

a) if Ti<<Tj then there is no path from Tj to Ti in TIO(s),
b) if Th<<Tj then Ti<<Tj, and
c) if Ti<<Tk then Ti<<Tj. ∎

Intuitively, TIO(s) has a DITS if the nodes can be linearly arranged horizontally in such a way that all arcs are directed from left to right and no two intervals "overlap". One of the virtues of the DITS is that it provides a uniform tool for dealing with many problems in the serializability theory.

Theorem 4.1. A schedule s is serializable iff TIO(s) has a DITS which orders To first and Tf last.

TIO(b) of Fig. 1 has two DITS's satisfying Theorem 4.1. They are
To<<T'o<<T3<<T'3<<T1<<T2<<T4<<T5<<Tf and
To<<T'o<<T1<<T2<<T4<<T3<<T'3<<T5<<Tf.
It follows easily from Theorem 4.1 that a schedule is not serializable if it contains two transactions T1 and T2 such that T1 (T2) reads a data item X (Y) from the same transaction and writes a new value of Y (X) (even if X = Y). This becomes clear if TIO(s) is drawn and Thoerem 4.1 is applied to it. Theorem 8 of [PAPA-79] follows easily from this observation.
As expected from the NP-completeness of serializability test [PAPA-79], testing the existence of a DITS for a transaction IO graph is in general NP-complete [IBAR-82].

5. Inclusion Relationship Among WW, WR, RW, etc

Since the membership test in SR is in general NP-complete, we impose some constraints on serialization order and want to test if the schedule is serializable under these additional constraints.

Definition 5.1. Let s = (OP(T), <s) be a schedule over a set T of transactions.

(a) [ww-constraints] If Wi(X)<sWj(X) for some data item X, then Ti must be serialized before Tj.
(b) [wr-constraints] If Wi(X)<sRj(X) for some X, then Ti must be serialized before Tj.
(c) [rw-constraints] If Ri(X)<sWj(X) for some X, then Ti must be serialized before Tj.
(d) [rr-constraints] If Ri(X)<sRj(X) for some X, then Ti must be serialized before Tj.

A schedule s belongs to the sets WW, WR, RW and RR, if s is serializable under conditions (a), (b), (c) and (d), respectively. ∎

In order to test if s ∈ WW, for example, we indicate in TIO(s) each ww-constraint by a dotted arc, called a ww-arc, from node Ti to node Tj. If the resulting graph is to have DITS with these constraints, then some additional constraints may be implied by them.

Definition 5.2. The conditions b) and c) of Definition 4.1 are refered to as the exclusion rule. Let (Th, Ti):X and (Tj, Tk):X be as defined in Definition 4.1. If there is a path in TIO(s) from Th to Tj, from Th to Tk, or from Ti to Tk, we introduce an unlabeled exclusion arc (Ti, Tj) induced by the exclusion rule. ∎

The term "exclusion rule" was used in [SETH-81] in a slightly different context, pertaining to individual operations instead of transactions. Suppose we add all ww-arcs to TIO(s). Then we repeatedly introduce exclusion arcs induced by the exclusion rule until the rule is no longer applicable. The resulting graph is said to be exclusion closed [SETH-81] and will be denoted by TIO[ww](s).

Theorem 5.1. Let c be any set of constraints that we have introduced above (ww, wr, etc.) and let C stand for the set of serializable schedules satisfying the constraints in c. A schedule s belongs to C if and only if TIO[c](s) has a DITS.

It can be proved [IBAR-82] that TIO(s) has a DITS satisfying the ww-constraints iff TIO[ww](s) is acyclic. However, the existence of a DITS under the constraints wr or rw cannot be tested by the acyclicity of TIO[wr](s) or TIO[rw](s).
We use the notation, WR+RW, for example, to denote the set of serializable schedules satisfying both the wr- and rw-constraints. Note that WR+RW⊊WR∩RW. To see this, refer to schedule b of Fig. 1 again. If TIO(b) is augmented by the wr-arcs (T3, T4), (To, T2) and (To, T4), it still has a DITS, and therefore b ∈ WR. Similarly, if TIO(b) is augmented by the rw-arcs, (T2, T3) and (T4, T5), then it has a DITS and therefore b ∈ RW. It follows that b ∈ WR∩RW. However, TIO(b) has no DITS if all these constraints are to be satisfied. We thus conclude that b ∉ WR+RW.
The membership in WR+RW can be tested in polynomial time (Section 6). However, the tests of membership in WR, WR+RR, RW, RW+RR, RR are all NP-complete, even for the "two-step" model [IBAR-82].
It turns out that WW = WW+WR+RW [IBAR-82]. In the two-step model, the set WW+WR+RW is called DSR [PAPA-79] or CPSR [BERN-79].
Finally, the inclusion relationship among all the classes defined above is summarized in

Fig. 2, where

a = Wo[X,Y]R1[X]R2[X]W2[X,Y]R3[X]W1[Y]W3[Y]Rf[X,Y],
b = Wo[X]W1[X]R2[X]W3[X]W2[X]R4[X]W5[X]Rf[X],
c = Wo[X]R2[X]R1[X]W2[X]Rf[X],
d = Wo[X]R3[X]W1[X]R2[X]W3[X]W2[X]Rf[X],
e = Wo[X,Y]R2[Y]R1[X]W2[X]W1[X]R3[X]W4[X]Rf[X,Y],
f = (To)d*(T1)e*(Tf), and
g = (To)d*(T1)e*(T2)c*(Tf).

In schedules f and g, T1 and T2 read and write all data items. The notation d*, for example, denotes the schedule obtained from d by stripping off the operations of its initial and final transactions, i.e., Wo[X] and Rf[X].

6. Polynomial Membership Test in WW and WR+RW

Lemma 6.1. Let c be a set of constraints and let C stand for the set of serializable schedules satisfying the constraints in c. We have $s \in C$ if and only if TIO[c](s) is acyclic, provided the following condition P holds.

Condition P: For each data item X and a pair Ti, Tj of transactions that write X, there is a path in TIO[c](s) either from Ti to Tj or from Tj to Ti, unless Wi[X] and Wj[X] are both useless writes, in which case such a path need not exist.

Theorem 6.1. Membership in WW and also in WR+RW can be tested in polynomial time.

It turns out that the TIO[ww](s) has a cycle iff the TIO(s) augmented by the ww-, wr- and rw-arcs has a cycle.

References

[BERN-79] Bernstein, P.A., Shipman, D.W., and Wong, W.S., Formal aspects of serializability in database concurrency control, IEEE Trans. Software Eng. SE-5, 3 (May 1979), 203-215.

[BERN-82] Bernstein, P.A. and Goodman, N., Multiversion concurrency control theory and algorithms, Proc. ACM SIGACT/SIGOPS Symp. on Distributed Computing, Aug. 1982, 209-215.

[ESWA-76] Eswaran, K.P., Gray, J.N., Lorie, R.A. and Traiger, I.L., The notions of consistency and predicate locks in a database system, Comm. ACM 19, 11 (Nov. 1976), 624-633.

[IBAR-82] Ibaraki, T., Kameda, T. and Minoura, T., SNOTS and its applications: serializability theory made simple, Tech Rept. 82-12, Dept. Comp. Sci., Simon Fraser University, 1982, 51pp.

[IBAR-83] Ibaraki, T. and Kameda, T., Manuscript under preparation, 1983.

[KATO-83] Kato, N. and Kameda, T., Manuscript under preparation, 1983.

[MURO-81] Muro, S., Minoura, T. and Kameda, T., Multi-version concurrency control for a database system, CCNG Rept. E-98, Computer Communications Networks Group, Univ. of Waterloo, Aug. 1981.

[PAPA-79] Papadimitriou, C.H., The serializability of concurrent database updates, JACM 26, 4 (Oct. 1979), 631-653.

[PAPA-82] Papadimitriou, C.H. and Kanellakis, P.C., On concurrency control by multiple versions, Proc. ACM Symp. on Principles of Database Systems, March 1982, 76-82.

[REED-79] Reed, D.P., Implementing atomic actions on decentralized data, Proc. 7th ACM Symp. Operating systems Principles, Dec. 1979, 66-74.

[SETH-81] Sethi, R., A model of concurrent database transactions, Proc. 22nd IEEE Symp. Foundation of Comp. Sci., Oct. 1981, 175-184.

[STEA-76] Stearns, R.E., Lewis, P.M. II, and Rosenkrantz, D.J., Concurrency control for database systems, Proc. 17th IEEE Symp. Foundation of Computer Sci., Oct. 1976, 19-32.

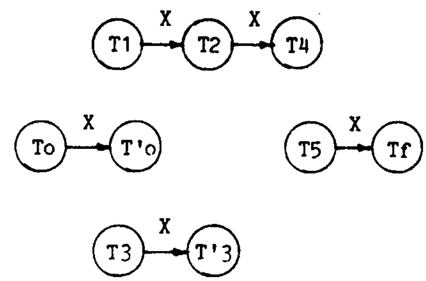

Fig. 1. TIO graph for b = Wo[X]W1[X]R2[X] W3[X]W2[X]R4[X]W5[X]Rf[X].

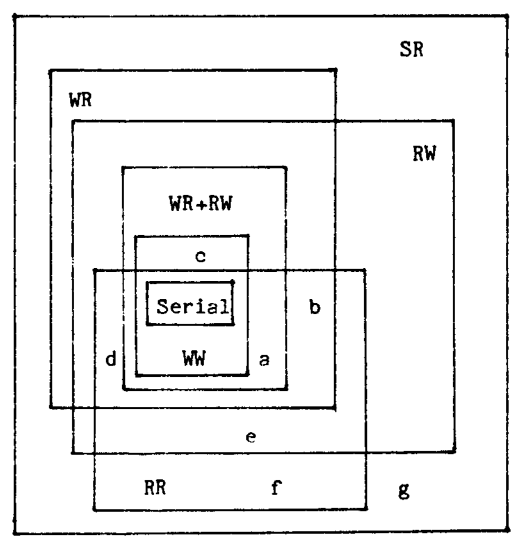

Fig. 2. Inclusion relationship.