International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

A Serializability Condition for Multi-step Transactions Accessing Ordered Data

Rafat Alshorman, Walter Hussak

Abstract—In mobile environments, unspecified numbers of transactions arrive in continuous streams. To prove correctness of their concurrent execution a method of modelling an infinite number of transactions is needed. Standard database techniques model fixed finite schedules of transactions. Lately, techniques based on temporal logic have been proposed as suitable for modelling infinite schedules. The drawback of these techniques is that proving the basic serializability correctness condition is impractical, as encoding (the absence of) conflict cyclicity within large sets of transactions results in prohibitively large temporal logic formulae. In this paper, we show that, under certain common assumptions on the graph structure of data items accessed by the transactions, conflict cyclicity need only be checked within all possible pairs of transactions. This results in formulae of considerably reduced size in any temporal-logic-based approach to proving serializability, and scales to arbitrary numbers of transactions.

Keywords—multi-step transactions, serializability, directed graph.

I. Introduction

MOBILE computing and the World Wide Web (WWW) have produced huge numbers of simultaneous users executing transactions in transaction processing systems. The scenario is one of unbounded numbers of transactions incoming and outgoing to databases in continuous streams and accessing common data items [1]. It has been noted [2], [3], [5] that modelling such systems, from the point of view of ascertaining such properties as correctness and absence of starvation, requires modelling infinite rather than the fixed finite schedules of classical database theory [4]. Infinite schedules can be modelled in temporal logic [2], [3], [5]. However, there is a significant drawback to modelling the main serializability correctness condition.

The problem with specifying serializability is with specifying the existence of cycles in the infinite conflict graph of a schedule or 'history' h. There are $n!$ ways that a cycle can occur between n transactions and an encoding into temporal logic results in a formula whose length is of order a factorial of the number of propositions that represent different active transactions at any point in time. In the case of a scheduler dealing with at most n active transactions at any point in time, exhaustive proofs of serializability are beyond even the most powerful model checkers available for any realistic value of n. The work [5] gives an encoding of serializability that is polynomial in such n, but uses a temporal logic of non-elementary computational complexity. In [2], a polynomial encoding into plain LTL is achieved for 2-step transactions. A much simpler encoding of serializability, that is more amenable to model checking, is given for multi-step transactions in [8], and requires cycles only to be checked between pairs of transactions. However, the assumption there is that all transactions access the same set of data items. In the general case, where transactions access different sets of data items, it is not sufficient to check for cycles between pairs of transactions. Consider the case of 3 transactions T_1, T_2 and T_3 which access the sets of data items $\{x, y\}$, $\{y, z\}$ and $\{x, z\}$ respectively, where x, y and z are all different, and the history h_1 whose steps occur in the following order:

$$\begin{aligned} h_1 &= r_3(x)w_3(x)r_1(x)w_1(x)r_1(y) \\ & w_1(y)r_2(y)w_2(y)r_2(z)w_2(z)r_3(z)w_3(z) \end{aligned}$$

Here, for example, $r_3(x)$ denotes transaction T_3 reading x, and $w_3(x)$ transaction T_3 writing to x. History h_1 has the following cycle of conflicts:

$$T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_3$$

(and is therefore not serializable). However, as can be checked, there is no cycle between two transactions. In fact, as the history h_2 below shows, the absence of a cycle between any $n-1$ transactions does not guarantee the absence of a cycle between n transactions:

$$\begin{aligned} h_2 &= r_n(x_1)w_n(x_1)r_1(x_1)w_1(x_1)r_1(x_2)w_1(x_2)\ldots \\ & \ldots r_i(x_i)w_i(x_i)r_i(x_{i+1})w_i(x_{i+1})\ldots \\ \ldots r_{n-1}&(x_{n-1})w_{n-1}(x_{n-1})r_{n-1}(x_n)w_{n-1}(x_n)r_n(x_n)w_n(x_n) \end{aligned}$$

Here,

$$T_1, \ldots, T_i, \ldots, T_n$$

access sets of data items

$$\{x_1, x_2\}, \ldots, \{x_i, x_{i+1}\} \; \ldots, \{x_n, x_1\}$$

where $x_1, \ldots, x_n$ are distinct. History h_2 has the cycle

$$T_n \rightarrow T_1 \rightarrow \ldots T_{n-1} \rightarrow T_n$$

but no cycle of fewer than n transactions - if T_i $(1 \leq i \leq n)$ is removed from the history, the (acyclic) order of conflicts is:

$$T_{i+1} \rightarrow \ldots \rightarrow T_n \rightarrow T_1 \rightarrow \ldots \rightarrow T_{i-1}$$

We notice that in the examples h_1 and h_2 above, it is impossible to define an order on the data items, so that each transaction only accesses sets of contiguous data items. In this paper, we show that if such an order exists, then

R. Alshorman is with the Department of computer science, Loughborough University, Loughborough, LE11 3TU, UK, e-mail: R.alshorman@lboro.ac.uk.

W. Hussak is with the Department of computer science, Loughborough University, Loughborough, LE11 3TU, UK, e-mail: W.Hussak@lboro.ac.uk.

Manuscript received April 30, 2009; revised May 11, 2009.

to check for conflict cyclicity within sets of transactions, it is sufficient to check for conflict cyclicity between pairs of transactions. This results in a test for serializability of concurrent multi-step transactions, similar to that in [8], but for an entirely different class of transactions where the sets of data items accessed by transactions need not be the same. This paper is structured as follows. First of all, in Section II, we give a model for infinite numbers of concurrent multi-step transactions accessing ordered data. In Section III, we prove that, in this model, a cycle exists in the (infinite) conflict graph if and only if a cycle exists between two transactions. A formal condition for serializability, that can be encoded into temporal logic, is given in Section IV. Applications that satisfy this model are discussed in Section V and concluding remarks are given in Section VI.

II. Concurrent multi-step transactions model

A. Histories

We shall denote the set of multi-step transactions by $T = \{T_i : i \in \mathbb{N}_1\}$, where $\mathbb{N}_1$ is the set of positive integers. A *history* (or *schedule*) h for the set T is an interleaved sequence of all the read and write steps of all the transactions in T such that the subsequence of h comprising the steps of T_i is exactly the sequence of steps of T_i occurring in the order that they do in T_i. We write $s_i <_h s_{i'}$ if step s_i of T_i occurs before step $s_{i'}$ of $T_{i'}$ in h. A read (respectively write) step, of transaction i on data item x, is denoted $r_i(x)$ (respectively $w_i(x)$). In the next definition, we define formally an order on the set of the data items from which data is accessed by transactions.

Definition 1. *Let $D = \{x_1, x_2, \ldots, x_m\}$ be an irreflexively totally ordered, by $<_D$ say, set of data items such that*

$$x_1 <_D \ldots <_D x_m$$

and $T = \{T_i : i \in \mathbb{N}_1\}$ be the set of transactions participating in history h. Denote by D_i the totally ordered set of data items accessed in turn by transaction T_i assumed to be of the form

$$D_i = \{x_a, x_{a+1}, \ldots, x_{b-1}, x_b\},$$

where $1 \le a \le b \le m$ and $D_i \subseteq D$. For the remainder of this paper, if a set of data items $D' \subseteq D$ is denoted by $\{x_a, \ldots, x_b\}$, this will mean that $x_a <_D \ldots <_D x_b$. We shall denote the case of transaction T_i preceding transaction $T_{i'}$ in accessing data items $x_p, \ldots, x_s$ in both read and write operations over history h, as $T_i <_h^{x_p,\ldots,x_s} T_{i'}$ where $x_p, x_{p+1}, \ldots, x_s \in D_i \cap D_{i'}$.

For example, assume that D_i and $D_{i'}$ are:

$$D_i = \{x_1, x_2, x_3\}$$

$$D_{i'} = \{x_2, x_3, x_4, x_5\}$$

where $x_1 <_D x_2 <_D x_3 <_D x_4 <_D x_5$. Then, in the history

$$h = \ldots r_i(x_2)w_i(x_2) \ldots r_i(x_3)w_i(x_3) \ldots r_{i'}(x_2)w_{i'}(x_2) \ldots r_{i'}(x_3)w_{i'}(x_3) \ldots$$

we have that $T_i <_h^{x_2,x_3} T_{i'}$.

B. Serializability

A *serial history* is a history in which all operations of any transaction T_i are executed consecutively in the history. Otherwise, the history is called *nonserial*. A 'serializable' history is a history h, that is 'equivalent' to some serial history of the same transactions. Various notions of equivalence have been defined as in, for example, [4] and [2] . In this paper, we shall adopt the common 'conflict equivalence'. Two histories are conflict equivalent if the order of any two 'conflicting' operations is the same in both histories. Two operations conflict if they belong to different transactions, access the same data item, and at least one of them is a write.

Definition 2. *Histories h_1 and h_2 of $T = \{T_i : i \in \mathbb{N}_1\}$ are* equivalent, *written as $h_1 \sim h_2$, iff for all $i, i' \ge 1, i \ne i'$, and for all $x \in D_i \cap D_{i'}$,*

1) *if $r_i(x) <_{h_1} w_{i'}(x)$, then $r_i(x) <_{h_2} w_{i'}(x)$ and*
2) *if $w_i(x) <_{h_1} w_{i'}(x)$, then $w_i(x) <_{h_2} w_{i'}(x)$*

Definition 3. *A history h of $T = \{T_i : i \in \mathbb{N}_1\}$ is* serializable *iff there is a serial history h_S of T of the form, for each $i \in \mathbb{N}_1$,*

$$h_S = \ldots \underbrace{\ldots r_i(x) \ldots w_i(y) \ldots}_{\text{only (all) steps of } T_i} \ldots$$

such that $h \sim h_S$.

C. Conflict graphs

'Conflict graphs' are widely used for testing conflict serializability of finite histories of transactions in polynomial time [4], [6], [7]. Definition 4, below, explains how we build the conflict graph for a history. Theorem 5 states that acyclicity of conflict graphs corresponds to serializability in the case of infinite histories.

Definition 4. *A* directed graph *is a pair $G = (V, A)$, where V is a set of elements called* nodes, *denoted nodes(G), and $A \subseteq V \times V$ is a set of elements called* arcs, *denoted arcs(G). A* walk *in a directed graph $G = (V, A)$ is a sequence of nodes $(v_1, v_2, \ldots, v_n)$ such that $(v_i, v_{i+1}) \in A$ for $i = 1, \ldots, n-1$. A walk with no nodes repeated is called a* path*; it is a* cycle *when only the first and last node coincide. For each history h, there is a directed graph $G(h)$ called the* conflict graph *of h. This graph has the transactions of h as its nodes, and contains an arc $(T_i, T_{i'})$, where T_i and $T_{i'}$ are distinct transactions of h, whenever there is a step of T_i which conflicts with a subsequent (in h) step of $T_{i'}$.*

Theorem 5. *A history h of an infinite number of multi-step transactions $T = \{T_i : i \in \mathbb{N}_1\}$, accessing data items in some finite set D, is serializable iff its conflict graph $G(h)$ is acyclic.*

Proof: This theorem is proved as Theorem 3.2 in [8]. ■

III. Cycle Reduction in Conflict Graphs

In this section, we prove a succession of properties of infinite histories culminating in Theorem 11. Theorem 11 is the main result and is used to give a condition for serializability in Section IV that can be encoded efficiently into common temporal logics such as CTL (Computational Tree Logic) or LTL (Linear-time Temporal Logic).

The first two lemmas, Lemma 6 and Lemma 7, give basic properties of the order $<_h^{x_p,\ldots,x_s}$ from Definition 1.

Lemma 6. *Let h be a history with no cycle of length 2 in its conflict graph, and suppose that $x \in D$ and transactions T_i and T_j are such that $x \in D_i \cap D_j$. Then,*

$$\text{either } T_i <_h^x T_j \text{ or } T_j <_h^x T_i \qquad (1)$$

Proof: By the definition of $<_h^x$ in Definition 1, the only way (1) can fail is if two reads $r_i(x)$ and $r_j(x)$ occur before the two writes $w_i(x)$ and $w_j(x)$. But, then the conflict graph would have the cycle (T_i, T_j), (T_j, T_i) of length 2. ■

Lemma 7. *Let h be a history with no cycle of length two in its conflict graph $G(h)$, $T_i <_h^{x_{a_1},x_{a_2}} T_j$, and suppose that x_a lies between x_{a_1} and x_{a_2}, i.e. $x_{a_1} <_D x_a <_D x_{a_2}$. Then, $T_i <_h^{x_a} T_j$, (see Figure 1).*

Proof: As $x_{a_1}, x_{a_2} \in D_i \cap D_j$, and as, by Definition 1, D_i and D_j contain contiguous elements, we have that $x_a \in D_i \cap D_j$. If $T_i <_h^{x_a} T_j$ does not hold then, by Lemma 6, we must have that $T_j <_h^{x_a} T_i$. This implies that there is an arc in $G(h)$ such that (T_j, T_i). As $T_i <_h^{x_{a_1}} T_j$, there is another arc (T_i, T_j) in $G(h)$ thereby completing a cycle of length 2 and contradicting the assumption that $G(h)$ has no cycle of length 2. ■

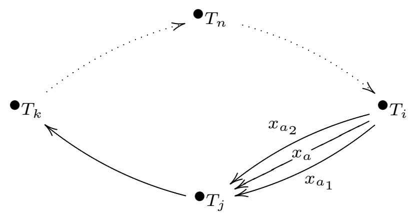


Fig. 1. The conflict graph $G(h)$ for Lemma 7.

For the next property Lemma 8 below, consider the case of three transactions T_i, T_j and T_k accessing data items such that the data items accessed by T_j straddle those accessed by T_i and T_k as follows:

$$D = \quad \{\overbrace{\underbrace{\ldots, x_{b'_1}, x_{b'_2}, \ldots, x_{b'_u}}_{T_k}, \ldots, \underbrace{x_{a'_1}, x_{a'_2}, \ldots, x_{a'_l}, \ldots}_{T_i}}^{T_j}\}$$

We claim that T_i, T_j, and T_k cannot be part of a cycle in $G(h)$ of length n where $n > 2$ if there is no cycle of length 2. For example, assume that there are 4 ordered sets of data items $D_i \supseteq \{x_a, x_c\}$, $D_j \supseteq \{x_b, x_a\}$, $D_k \supseteq \{x_z, x_b\}$ and $D_l \supseteq \{x_z, x_c\}$ accessed by corresponding transactions T_i, T_j, T_k and T_l respectively, which form a cycle as in Figure 2 below. Since D_l contains x_z and x_c it follows, by Definition 1, that x_b and x_a are also in D_l, i.e.,

$$D_l \supseteq \{x_z, x_b, x_a, x_c\}.$$

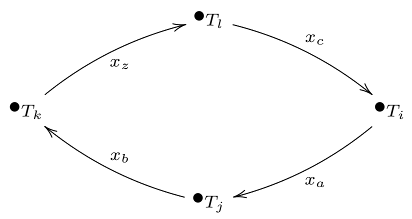
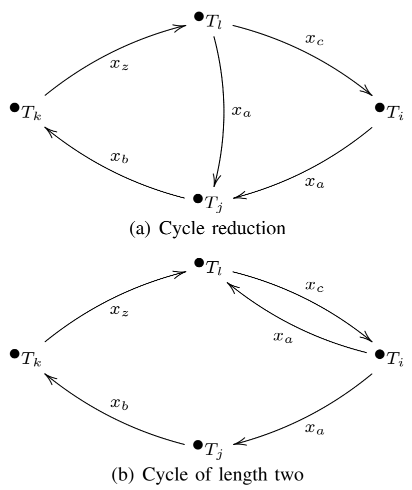


Fig. 2. The conflict graph $G(h)$ which contains T_i, T_j, T_k and T_l.

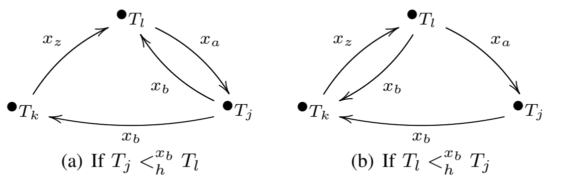
Thus, $x_a \in D_i \cap D_l$ and so, by Lemma 6, either $T_i <_h^{x_a} T_l$ or $T_l <_h^{x_a} T_i$. If $T_l <_h^{x_a} T_i$ then, by transitivity of $<_h^{x_a}$, we can reduce the cycle, as in Figure 3(a). But, if $T_i <_h^{x_a} T_l$ then, we have a cycle of length two; see Figure 3(b). Now, for the case of Figure 3(a), from above we have that $x_b \in D_j \cap D_l$. Therefore, by Lemma 6, either $T_j <_h^{x_b} T_l$ or $T_l <_h^{x_b} T_j$. If $T_l <_h^{x_b} T_j$ then, by transitivity, we have a cycle of length two; see Figure 4(b). But, if $T_j <_h^{x_b} T_l$, then we also have a cycle of length two, see Figure 4(a). Below, Lemma 8 shows that in fact our claim is true for $n \geq 3$ number of transactions. Lemma 8 will be used to prove Lemma 9.



(a) Cycle reduction

(b) Cycle of length two

Fig. 3. We can either reduce the cycle or make a cycle of length two.



(a) If $T_j <_h^{x_b} T_l$ (b) If $T_l <_h^{x_b} T_j$

Fig. 4. We have a cycle of length two if either $T_j <_h^{x_b} T_l$ or $T_l <_h^{x_b} T_j$

Lemma 8. *Suppose that D_i, D_j and D_k are the sets of data items accessed by T_i, T_j and T_k, respectively, and are of the form*

$$D_i = \{x_{a'_1}, x_{a'_2}, \ldots, x_{a'_l}, \ldots, x_c, \ldots\}$$
$$D_j = \{\ldots, x_{b'_1}, x_{b'_2}, \ldots, x_{b'_u}, \ldots, x_{a'_1}, x_{a'_2}, \ldots, x_{a'_l}, \ldots\}$$

$$D_k = \{\dots, x_{b'_1}, x_{b'_2}, \dots, x_{b'_u}\}$$

where, for all $x_{a'} \in \{x_{a'_i} : 1 \le i \le l\}$ and $x_{b'} \in \{x_{b'_g} : 1 \le g \le u\}$, $x_{b'} <_D x_{a'}$, $x_c \notin D_j$, $T_i <_h^{x_{a'}} T_j$ and $T_j <_h^{x_{b'}} T_k$. Then, there is no cycle in $G(h)$ of minimum length n, where $n > 2$ is of the form:

$$(T_i, T_j), (T_j, T_k), \dots, (T_l, T_{l+1}), \dots, (T_s, T_i)$$

and $T_s <_h^{x_c} T_i$.

Proof: Assume, on the contrary, that we have a cycle of minimum length $n > 2$,

$$(T_i, T_j), (T_j, T_k), (T_1, T_2), \dots, (T_s, T_i) \quad (s \ge 0) \qquad (2)$$

(where $s = 0$ is the case $T_s = T_k$). Let $x_0 = x_{b'}$, x_1, ... , $x_s = x_c$, $x_{s+1} = x_{a'}$ be such that, putting $T_i = T_{s+1}, T_j = T_{s+2}$, and $T_k = T_0$:

$$T_{s+1}(= T_i) <_h^{x_{a'}} T_{s+2}(= T_j),\ T_{s+2}(= T_j) <_h^{x_{b'}} T_0(= T_k), \dots$$
$$\dots,\ T_l <_h^{x_l} T_{l+1}, \dots,\ T_s <_h^{x_s} T_{s+1}(= T_i)$$

We have that $x_c >_D x_{a'} >_D x_{b'}$, but, clearly, we cannot have

$$x_{a'} >_D x_{b'} >_D x_1 >_D \dots >_D x_l >_D \dots >_D x_c >_D x_{a'}$$

Therefore, there is some l, with $0 \le l \le s-1$, such that

$$x_l <_D x_{l+1} >_D x_{l+2} \qquad (3)$$

There are two cases to consider corresponding to (3):

Case (i) $x_{l+2} <_D x_l <_D x_{l+1}$. In this case, as

$$x_{l+2} <_D x_l <_D x_{l+1},$$

and

$$T_l <_h^{x_l} T_{l+1} <_h^{x_{l+1}} T_{l+2} <_h^{x_{l+2}} T_{l+3}$$

we have that x_{l+1}, x_{l+2} and therefore x_l belong to T_{l+2}. As no cycle of length 2 exists between T_{l+1} and T_{l+2}, we must have that $T_{l+1} <_h^{x_l} T_{l+2}$. Therefore, as $T_l <_h^{x_l} T_{l+1}$, by transitivity we have that $T_l <_h^{x_l} T_{l+2}$. This produces a cycle reduction of (2), which is a contradiction.

Case (ii) $x_l <_D x_{l+2} <_D x_{l+1}$. In this case, as

$$x_l <_D x_{l+2} <_D x_{l+1},$$

and

$$T_l <_h^{x_l} T_{l+1} <_h^{x_{l+1}} T_{l+2} <_h^{x_{l+2}} T_{l+3}$$

we have that x_l, x_{l+1} and therefore x_{l+2} belong to T_{l+1}. As no cycle of length 2 exists between T_{l+1} and T_{l+2}, we must have that $T_{l+1} <_h^{x_{l+2}} T_{l+2}$. By transitivity, from $T_{l+1} <_h^{x_{l+2}} T_{l+2}$ and $T_{l+2} <_h^{x_{l+2}} T_{l+3}$, we get $T_{l+1} <_h^{x_{l+2}} T_{l+3}$ giving a cycle reduction. This contradiction completes the proof of the lemma. ∎

Now, consider the case where $z <_D y <_D x$ and T_1, T_2 and T_3 access the following sets of data items

$$D_1 = \{x\}, D_2 = \{y, x\} \text{ and } D_3 = \{z, y\}.$$

Also, consider a history h, which contains T_1, T_2 and T_3, of the form:

$$h = \ \dots r_2(y) \dots w_2(y) \dots r_3(y) \dots w_3(y) \dots r_1(x) \dots w_1(x)$$
$$\dots r_2(x) \dots w_2(x) \dots$$

The corresponding conflict graph $G(h)$, for the history h, is shown in Figure 5. This shows a situation that Lemma 9 can remove, i.e. Lemma 9 asserts that if we have cycle in $G(h)$ of length n, where $n > 2$, then, any three consecutive transactions T_i, T_j and T_k, participating in $G(h)$, should contain $x_{a'}$ and $x_{b'}$ such that $x_{a'} <_D x_{b'}$, $T_i <_h^{x_{a'}} T_j$ and $T_j <_h^{x_{b'}} T_k$. We need such $x_{a'}$ and $x_{b'}$ of Lemma 9 along with Lemma 10 to prove the main result Theorem 11 that reduces to cycles of length 2.

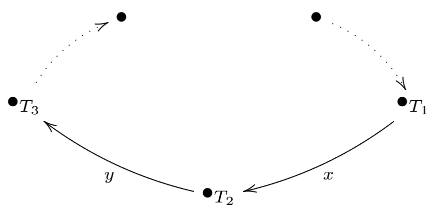


Fig. 5. No cycle in $G(h)$

Lemma 9. *Let h be a history with a cycle in $G(h)$ of minimum length n, where $n > 2$, containing arcs (T_i, T_j) and (T_j, T_k), and x_a, x_b such that $T_i <_h^{x_a} T_j$ and $T_j <_h^{x_b} T_k$, Then, there exist $x_{a'}, x_{b'}$ such that $x_{a'} <_D x_{b'}$ and $T_i <_h^{x_{a'}} T_j$ and $T_j <_h^{x_{b'}} T_k$.*

Proof: Assume, on the contrary, that we have a cycle in $G(h)$ of length n, where $n > 2$, containing arcs $(T_i, T_j), (T_j, T_k)$ such that $T_i <_h^{x_a} T_j$ and $T_j <_h^{x_b} T_k$ for some x_a, x_b, but that there are no $x_{a'}, x_{b'}$ such that $x_{a'} <_D x_{b'}$ and $T_i <_h^{x_{a'}} T_j$ and $T_j <_h^{x_{b'}} T_k$. This means that, for all $x_{a'} \in D_i \cap D_j$ such that $T_i <_h^{x_{a'}} T_j$, and, for all $x_{b'} \in D_j \cap D_k$ such that $T_j <_h^{x_{b'}} T_k$, we have that $x_{b'} <_D x_{a'}$. Therefore, D_i contains data items $x_{a'} \in D_i \cap D_j = \{x_{a'_1}, x_{a'_2}, \dots, x_{a'_l}\}$, D_k contains data items $x_{b'} \in D_j \cap D_k = \{x_{b'_1}, x_{b'_2}, \dots, x_{b'_u}\}$ and D_j contains all data items $x_{a'} \in D_i \cap D_j$ and $x_{b'} \in D_j \cap D_k$ so that $D_j \supseteq \{x_{b'_1}, x_{b'_2}, \dots, x_{b'_u}, \dots, x_{a'_1}, x_{a'_2}, \dots, x_{a'_l}\}$, i.e. D_j is of the form:

$$D_j = \{\dots, x_{b'_1}, \dots, x_{b'_u}, \dots, x_{a'_1}, \dots, x_{a'_l}, \dots\} \qquad (4)$$

We show, from (4), that, in fact, D_i and D_k should be as follows

$$D_i = \{x_{a'_1}, x_{a'_2}, \dots, x_{a'_l}, \dots\} \text{ and} \qquad (5)$$
$$D_k = \{\dots, x_{b'_1}, x_{b'_2}, \dots, x_{b'_u}\} \qquad (6)$$

Firstly, assume that (5) does not hold, i.e we can find $x_c \in D_i$ such that $x_c <_D x_{a'_1}$ and

$$D_i = \{\underbrace{\dots\dots}_{x_c}, x_{a'_1}, \dots, x_{a'_l}, \dots\}.$$

From (4), we can choose x_c to be such that $x_c \in D_j$ and therefore $x_c \in D_i \cap D_j$. If x_c is such that $T_i <_h^{x_c} T_j$ then,

$x_c \in \{x_{a'_i} : 1 \le i \le l\}$ and this contradicts our assumption that $x_c <_D x_{a'_1}$. On the other hand, if $T_j <_h^{x_c} T_i$, we have (T_j, T_i) in $G(h)$ which with (T_i, T_j), from $T_i <_h^{x_a} T_j$, completes a cycle of length 2 which contradicts the hypothesis of this lemma. Thus, we have now shown that (5) must hold.

Secondly, assume that (6) does not hold, i.e we can find x_c such that $x_{b'_u} <_D x_c$ and $x_c \in D_k \cap D_j$. If x_c is such that $T_j <_h^{x_c} T_k$ then, $x_c \in \{x_{b'_i} : 1 \le i \le u\}$ and this contradicts our assumption that $x_{b'_u} <_D x_c$. On the other hand, if $T_k <_h^{x_c} T_j$, we have an arc (T_k, T_j) in $G(h)$ which, along with the arc (T_j, T_k), forms a cycle of length 2 contradicting the hypothesis of this lemma. Thus (6) must hold.

We shall now show that our main assumption that no $x_{a'} <_D x_{b'}$ exists such that $T_i <_h^{x_{a'}} T_j$ and $T_j <_h^{x_{b'}} T_k$, leads to a contradiction. Since, we have a cycle, there is x_c and a transaction T_{i-1} such that $T_{i-1} <_h^{x_c} T_i$. We cannot choose such x_c in $\{x_{a'_i} : 1 \le i \le l\}$ because then we could reduce the cycle as we would have $x_c \in T_j$ and therefore $T_{i-1} <_h^{x_c} T_j$ and an arc (T_{i-1}, T_j). Also, we cannot choose x_c such that $x_{a'_l} <_D x_c$ and $x_c \in D_j$ as we could reduce the cycle because then $T_{i-1} <_h^{x_c} T_j$. However, if $x_{a'_l} <_D x_c$ and $x_c \notin D_j$, we have that

$$D_i = \{x_{a'_1}, x_{a'_2}, \ldots, x_{a'_l}, \ldots, x_c, \ldots\}$$

and $x_c \notin D_j$ giving the conditions of Lemma 8. Application of Lemma 8 shows that (T_i, T_j) and (T_j, T_k) could not form part of a cycle. This contradiction completes the proof. ■

Lemma 10. *If h is a history with a cycle in $G(h)$ of minimum length n, where $n > 2$, and there are T_i, T_j and T_k such that $T_i <_h^{x_{a'}} T_j$, $T_j <_h^{x_{b'}, x_{b'_1}} T_k$, and $x_{a'} <_D x_{b'}$. Then, $x_{a'} <_D x_{b'_1}$, as in Figure 6.*

Proof: Assume, on the contrary, that $x_{b'_1} <_D x_{a'}$. As $x_{a'} <_D x_{b'}$, then $x_{a'}, x_{b'}$ and $x_{b'_1}$ will be ordered such that

$$x_{b'_1} <_D x_{a'} <_D x_{b'} \tag{7}$$

As $x_{b'} \in D_j \cap D_k$ and $x_{b'_1} \in D_j \cap D_k$ then, by Lemma 7 and (7), $x_{a'} \in D_j \cap D_k$. Therefore, we should have either $T_j <_h^{x_{a'}} T_k$ or $T_k <_h^{x_{a'}} T_j$. Now, if $T_j <_h^{x_{a'}} T_k$, then we can reduce the cycle via $T_i <_h^{x_{a'}} T_k$ giving a contradiction. But, if $T_k <_h^{x_{a'}} T_j$, then this gives an arc (T_k, T_j) which with the arc (T_j, T_k) from $T_j <_h^{x_{b'}} T_k$, completes a cycle of length 2 which is also a contradiction.

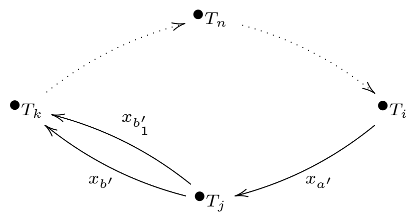


Fig. 6. Represents Lemma 10

■

From the previous lemma, we conclude that if we have the data item $x_a \in D$ such that $T_i <_h^{x_a} T_j$, $\{x_{b_i} : 1 \le i \le u\} \subseteq D$ such that $T_j <_h^{x_{b_1}, \ldots, x_{b_u}} T_k$, and there exists x_b such that $x_a <_D x_b$, where $x_b \in \{x_{b_i} : 1 \le i \le u\}$, then each data item $x_{b'}$ in $\{x_{b_i} : 1 \le i \le u\}$ will be such that $x_a <_D x_{b'}$. This will be used in the following Theorem 11.

Theorem 11. *Let h be a history over transactions $T = \{T_i : i \in \mathbb{N}_1\}$, where each $T_i \in T$ accesses the data items $D_i \subseteq D$. Then, if $G(h)$ has a cycle of length n and $n \ge 3$, there are two transactions T_{i_1}, T_{i_2} such that $G(h)$ has the cycle (T_{i_1}, T_{i_2}),(T_{i_2}, T_{i_1}).*

Proof: Assume that $G(h)$ has a cycle

$$(T_1, T_2), (T_2, T_3), \ldots, (T_n, T_1) \tag{8}$$

but no such cycle between two transactions. Choose $x_a, x_b, x_c, \ldots, x_d$ such that:

$$T_1 <_h^{x_a} T_2, T_2 <_h^{x_b} T_3, T_3 <_h^{x_c} T_4, \ldots, T_n <_h^{x_d} T_1$$

(see Figure 7). Put:

$$x_{a''} = min\{x_{a'_i} \in D : T_1 <_D^{x_{a'_i}} T_2\}$$

$$x_{b''} = min\{x_{b'_i} \in D : T_2 <_D^{x_{b'_i}} T_3\}$$

$$x_{c''} = min\{x_{c'_i} \in D : T_3 <_D^{x_{c'_i}} T_4\}$$

$$\ldots$$

$$x_{d''} = min\{x_{d'_i} \in D : T_n <_D^{x_{d'_i}} T_1\}$$

(see Figure 8), where *min* is with respect to $<_D$. By Lemma 9, there exist $x_{a'}$ and $x_{b'}$ such that:

$$x_{a'} <_D x_{b'},\ T_1 <_h^{x_{a'}} T_2,\ T_2 <_h^{x_{b'}} T_3$$

By Lemma 10, as $T_1 <_h^{x_{a'}} T_2$, $T_2 <_h^{x_{b'}, x_{b''}} T_3$ and $x_{a'} <_D x_{b'}$, we have that $x_{a'} <_D x_{b''}$. As, by the definition of $x_{a''}$, $x_{a''} \le_D x_{a'}$, we have that $x_{a''} \le_D x_{a'} <_D x_{b''}$ and so

$$x_{a''} <_D x_{b''} \tag{9}$$

In a similar fashion, we can prove that:

$$x_{b''} <_D x_{c''}, \ldots, x_{d''} <_D x_{a''} \tag{10}$$

From (9) and (10) we have that

$$x_{a''} <_D x_{b''}, \ldots, x_{d''} <_D x_{a''}$$

The contradiction $x_{a''} <_D x_{a''}$ completes the proof. ■

Theorem 11 shows that if we have a cycle in the conflict graph $G(h)$ of length n, where $n > 2$, then there is a cycle of length two in $G(h)$. This result is the basis of the serializability condition for these kinds of transactions, given in Theorem 12, below.

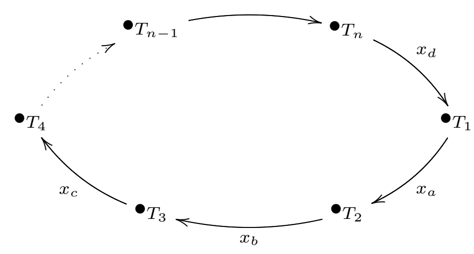


Fig. 7. Cycle in $G(h)$ of length n

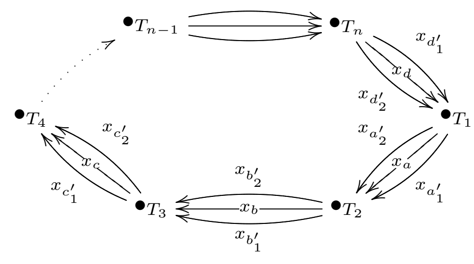


Fig. 8. $G(h)$, as in Figure 7, after applying Lemma 9

IV. The Serializability Condition

Theorem 12 gives a condition for testing for cycles between two transactions in a history and hence, by Theorem 5 and Theorem 11, a test for serializability. The condition can easily be expressed in either of the temporal logics CTL or LTL.

Theorem 12. *A history h of transactions $T = \{T_i : i \in \mathbb{N}_1\}$ is serializable iff for any two transactions $T_i, T_{i'} \in T$ $(i, i' \geq 1, i \neq i')$ one of them, T_i say, is such that, for all $x \in D_i \cap D_{i'}$, $w_i(x) <_h r_{i'}(x)$.*

Proof: Only if part
Assume that h is a history where the condition is not satisfied. One possibility is that there exists a data item x_1 such that $r_{i'}(x_1) <_h w_i(x_1)$, giving an arc $(T_{i'}, T_i)$ in $G(h)$, and another different data item x_2 such that $w_i(x_2) <_h r_{i'}(x_2)$, giving an arc $(T_i, T_{i'})$ in $G(h)$ thereby completing a cycle. Hence, in this case, by Theorem 5, h is not serializable. The other possibility is that the condition is breached on a single data item, i.e. one of the following cases:

$$\ldots r_i(x) \ldots r_{i'}(x) \ldots w_i(x) \ldots w_{i'}(x) \ldots \tag{11}$$
$$\ldots r_{i'}(x) \ldots r_i(x) \ldots w_i(x) \ldots w_{i'}(x) \ldots \tag{12}$$
$$\ldots r_i(x) \ldots r_{i'}(x) \ldots w_{i'}(x) \ldots w_i(x) \ldots \tag{13}$$
$$\ldots r_{i'}(x) \ldots r_i(x) \ldots w_{i'}(x) \ldots w_i(x) \ldots \tag{14}$$

In cases (11)-(14), it is clear that the conflict graphs of the histories are cyclic and, by Theorem 5, not serializable.
If part
Assume that the history h is not serializable. We show that the condition does not hold. To say that h is not serializable means, by Theorem 5, that there exists a cycle in $G(h)$. From Theorem 11, $G(h)$ has a cycle $(T_i,T_{i'})$,$(T_{i'},T_i)$, where $i, i' \geq 1$ and $i \neq i'$. Then, h is one of the following forms:

$$\ldots r_i(x) \ldots \underline{r_{i'}(x)} \ldots \underline{w_i(x)} \ldots w_{i'}(x) \ldots \tag{15}$$
$$\ldots r_i(x) \ldots \underline{r_{i'}(x)} \ldots w_{i'}(x) \ldots \underline{w_i(x)} \ldots \tag{16}$$
$$\ldots \underline{r_{i'}(x)} \ldots r_i(x) \ldots w_{i'}(x) \ldots \underline{w_i(x)} \ldots \tag{17}$$
$$\ldots \underline{r_{i'}(x)} \ldots r_i(x) \ldots \underline{w_i(x)} \ldots w_{i'}(x) \ldots \tag{18}$$
$$\ldots r_i(x) \ldots w_{i'}(x) \ldots \underline{r_{i'}(y)} \ldots \underline{w_i(y)} \ldots \tag{19}$$
$$\ldots r_{i'}(x) \ldots w_i(x) \ldots \underline{r_i(y)} \ldots \underline{w_{i'}(y)} \ldots \tag{20}$$

In (15)-(20) steps are underlined if they cause the condition to be breached. Cases (15)-(18) are when one date item x causes a cycle, and cases (19) and (20) are when 2 data items x and y cause a cycle. ■

Definition 13. *If $T' \subseteq T$, then the* projection *of h to T', denoted $h_{T'}$, is the history of T', obtained from h, by deleting all steps of transactions not in T'.*

To explain how the serializability condition of Theorem 12 is used to verify whether a history h is serializable or not, we shall give the following example:
Let T_1, T_2, T_3 and T_4 be multi-step transactions as follows

$$T_1 = r_1(x_2)w_1(x_2)r_1(x_3)w_1(x_3)r_1(x_4)w_1(x_4)$$
$$T_2 = r_2(x_1)w_2(x_1)r_2(x_2)w_2(x_2)$$
$$T_3 = r_3(x_2)w_3(x_2)r_3(x_3)w_3(x_3)r_3(x_4)w_3(x_4)r_3(x_5)w_3(x_5)$$
$$T_4 = r_4(x_2)w_4(x_2)r_4(x_3)w_4(x_3).$$

Let D be the set of all data items as follows

$$D = \{x_1, x_2, x_3, x_4, x_5\}$$

Also, let $h_{T'}$ be the history of $T' = \{T_i : 1 \leq i \leq 4\}$, where $T' \subseteq T$,

$$\begin{aligned} h_{T'} = {} & r_1(x_2)r_2(x_1)w_2(x_1)w_1(x_2)r_2(x_2)r_3(x_2)w_3(x_2) \\ & w_2(x_2)r_4(x_2)w_4(x_2)r_1(x_3)w_1(x_3)r_3(x_3)w_3(x_3) \\ & r_4(x_3)w_4(x_3)r_1(x_4)w_1(x_4)r_3(x_4)w_3(x_4)r_3(x_5) \\ & w_3(x_5) \end{aligned}$$

Firstly, we shall chop the history $h_{T'}$ up into sets of histories each containing two different transactions. Then, we shall check whether the serializability condition is satisfied for each set, to see if the history $h_{T'}$ is serializable. If $h_{T'}$ is not serializable, the main history h will not be serializable. Consider:

$$\begin{aligned} h_{\{T_1,T_2\}} = {} & r_1(x_2)r_2(x_1)w_2(x_1)w_1(x_2)r_2(x_2)w_2(x_2)r_1(x_3) \\ & w_1(x_3)r_1(x_4)w_1(x_4). \end{aligned}$$

We notice, from $h_{\{T_1,T_2\}}$, that $D_1 \cap D_2 = \{x_2\}$. According to the serializability condition in Theorem 12, if h is serializable then, we should have either, for all $x \in D_1 \cap D_2$, $w_1(x) <_h r_2(x)$ or, for all $x \in D_1 \cap D_2$, $w_2(x) <_h r_1(x)$. From $h_{\{T_1,T_2\}}$, we have $w_1(x_2) <_h r_2(x_2)$. This means that T_1 and T_2 satisfy the condition. Next, consider:

$$\begin{aligned} h_{\{T_1,T_3\}} = {} & r_1(x_2)w_1(x_2)r_3(x_2)w_3(x_2)r_1(x_3)w_1(x_3)r_3(x_3) \\ & w_3(x_3)r_1(x_4)w_1(x_4)r_3(x_4)w_3(x_4)r_3(x_5)w_3(x_5) \end{aligned}$$

From $h_{\{T_1,T_3\}}$, we notice that $D_1 \cap D_3 = \{x_2, x_3, x_4\}$. Also, we have for all $x \in D_1 \cap D_3$, $w_1(x) <_h r_3(x)$. This means that T_1 and T_3 satisfy the condition. Next, consider:

$$h_{\{T_1,T_4\}} = r_1(x_2)w_1(x_2)r_4(x_2)w_4(x_2)r_1(x_3)w_1(x_3)r_4(x_3) \\ w_4(x_3)r_1(x_4)w_1(x_4).$$

In $h_{\{T_1,T_4\}}$, we have, for all $x \in D_1 \cap D_4 = \{x_2, x_3\}$, $w_1(x) <_h r_4(x)$. This means that T_1 and T_4 satisfy the condition. Next, consider:

$$h_{\{T_2,T_3\}} = r_2(x_1)w_2(x_1)r_2(x_2)r_3(x_2)w_3(x_2)w_2(x_2)r_3(x_3) \\ w_3(x_3)r_3(x_4)w_3(x_4)r_3(x_5)w_3(x_5).$$

In $h_{\{T_2,T_3\}}$, we have $r_2(x_2) <_h w_3(x_2)$ and $r_3(x_2) <_h w_2(x_2)$. This breaches the condition and therefore history h is not serializable; see Figure 9. As h is not serializable, there is no point in checking the serializability condition for the remaining $h_{\{T_2,T_4\}}$ and $h_{\{T_3,T_4\}}$.

$$h_{\{T_2,T_4\}} = r_2(x_1)w_2(x_1)r_2(x_2)w_2(x_2)r_4(x_2)w_4(x_2) \\ r_4(x_3)w_4(x_3).$$

$$h_{\{T_3,T_4\}} = r_3(x_2)w_3(x_2)r_4(x_2)w_4(x_2)r_3(x_3)w_3(x_3)r_4(x_3) \\ w_4(x_3)r_3(x_4)w_3(x_4)r_3(x_5)w_3(x_5).$$

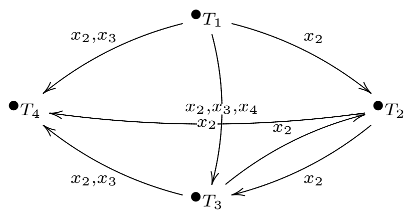


Fig. 9. $G(h_{T'})$ is a subgraph of $G(h)$.

V. Applications

Dividing transactions into sets of steps to produce multi-step transactions improves system throughput allowing the interleaved transactions to gain more parallelism [1]. Examples of multi-step transactions are when users enter data using a sequence of forms. At the end of the sequence, the application performs updates corresponding to input data. Desktop applications using wizards are a simplification, for the user, of the same process. In e-commerce sites, checkout flow can be seen as a multi-step transaction. Booking e-tickets from travel agencies also involves executing multi-step transactions. Ticket booking from the web (or e-ticketing for events, amusements, bus or flight tickets) is one of the widely available services in E-commerce. Customers can access the database and book a ticket at any time in any location.

The scenario of booking tickets is interesting in that the list of destinations are naturally ordered. It involves browsing the list of destinations, then checking the availability of seats, and booking one or more of them consecutively. For example, assume a passenger intends to book a ticket from location A to E. Firstly, he/she has to browse the list of available destinations from A. Secondly, choose the itinerary (the set of destinations may transit during the journey). Finally, book the itinerary. This scenario can be implemented as a multi-step transaction accessing ordered data, where a read step corresponds to browsing journey times from a destination, and the write step represents the booking phase of a chosen time to the next destination in the order. Assume that D represents the set of available destinations in the travel agency starting from A and ending at E, and $x_i \in D$ represents the next leg of the journey from i, such that $x_i \in D$. The ordered set of data items is depicted in Figure 10. The widespread use of e-tickets makes the number of incoming and outgoing transactions of unknown even though, at any point in time, the number of active transactions in the web server is finite.

$$\begin{tikzcd} \bullet_A \arrow[r] & \bullet_B \arrow[r] & \bullet_C \arrow[r, dotted] & \bullet_i \arrow[r, "x_i"'] & \bullet \arrow[r, dotted] & \bullet_E \end{tikzcd}$$

Fig. 10. Ordered set

VI. Conclusion

The emergence of mobile computing and the World Wide Web (WWW) has resulted in an indeterminate numbers of users expecting to execute their database transactions concurrently. In this paper, we have provided a serializability condition that can be used to verify the correctness of infinite histories that can model such large numbers of transactions in the case where the data accessed is ordered. The main advantage of the serializability condition which has been given, is that the testing for serializability only requires considering pairs of transactions. This makes testing for serializability efficient and easy to encode into the widely used temporal logics CTL and LTL. The modelling and verification process is one of specifying a scheduler by a transition system, encoding the serializability condition given here in temporal logic (either CTL and LTL) and then running a model checker to (automatically) perform the verification that the scheduler satisfies the serializability condition in all executions. This can all be done using common model checkers such as NuSMV and SPIN.

Further work will look to define a serializability condition for infinite histories of concurrent multi-step transactions accessing sets of data items with different graph properties which have other applications in the real-world.

Acknowledgment

We would like to thank ZPU (Zarqa Private University) and Loughborough University for their grant in making this work possible.

References

[1] R. Alshorman and W. Hussak, *Multi-step transactions specification and verification in a mobile database community*. In 3rd IEEE International Conference on Information Technologies: from Theory to Applications, IEEE, ICTTA 08, Damacus, Syria, IEEE Computer Society Press, 2008, pp. 1407-12.

[2] W. Hussak, *Specifying strict serializability of iterated transactions in Propositional Temporal Logic*, International Journal of Computer Science, vol. 2, issue 2 (2007), pp. 150-156.

[3] W. Hussak, *The serializability problem for a temporal logic of transaction queries*, Journal of Applied Non-Classical Logics, vol. 18, issue 1 (2008), pp. 67-78.

[4] C.H. Papadimitriou, *The Theory of Database Concurrency Control*, Computer Science Press, Pockville, Maryland, 1986.

[5] W. Hussak, *Serializable histories in Quantified Propositional Temporal Logic*, International Journal of Computer Mathematics, vol. 81, issue 10 (2004), pp. 1203-1211.

[6] R. Elmasri and S. Navathe, *Fundamental of Database Systems*. Addison-Wesley, Fourth Edition, 2004.

[7] A. Philip Bernstein, Vassos Hadzilacos and Nathan Goodman: *Concurrency Control and Recovery in Database Systems*. Addison Wesley Publishing Company, 1987.

[8] R. Alshorman and W. Hussak, *Computational Tree Logics for specifying multi-step transactions*, April 2009, Internal Report, No. 1102.