Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 40(1), 2013, Pages 1–8
ISSN: 1223-6934

Reducing conflict graph of multi-step transactions accessing ordered data with gaps

Rafat Alshorman and Hamed Fawareh

Abstract. Specifying the correctness of concurrent transactions competing to access a database is a crucial work. In this research, we adopt the serializability to be the correctness criterion for concurrent transactions accessing ordered data items. One of the most popular technique uses to test serializability is conflict graph. We add a realistic condition on the data items to show that the conflict graph cyclicity can be reduced, in a way, to use temporal-logic-based verification approach in proving serializability.

2010 Mathematics Subject Classification. Database theory 68P15; Directed graphs 05C20.
Key words and phrases. Serializability, conflict graphs, cycle reduction.

1. Introduction and Previous Works

Usually, database techniques are scheduling a finite number of transactions [8], [7]. Recently, with the emergence of new techniques such as web transactions and mobile databases, modeling and scheduling unlimited number of transactions, which may be representing infinite histories, has been recognized [4], [6], [5]. Representing infinite histories of iterative transactions can be performed using temporal logic formulas. The availability of powerful model checkers such as NuSMV [1] and SPIN [11] gives temporal logic formulas impetus and significance. Model checkers can carry out exhaustive checks of a correctness condition of serializability, and are fully automatic [2]. This means that no special expertise in mathematical proofs is required to carry out such verification. In this paper, we add a realistic condition on the set of data items. This addition makes testing and specifying of serializability more computationally efficient and applicable [3]. Specifying the correctness of concurrent transactions can be performed using temporal logics such as CTL (Computational Tree Logic) and LTL (Linear-time Temporal Logic). The serializability is based on acyclicity of conflict graphs. In order to be able to use such a correctness condition, we consider that acyclicity of conflict graphs corresponds to serializability for infinite schedules [10]. Then, we assume the further property for our transactions, that they access ordered set of data items with 'gaps'. We show that serializability that corresponds to the efficient condition where only cycles of length n have to be tested, and this condition is used for temporal logics specification. This work advances that of [5] and [4], which both deal with two-step transactions, to the more normal case of multi-step transactions. Moreover, in [9] and [10] they assumed that all transactions access the same set of data items or access different sets of data items where each transaction accesses consecutive data items in order. In this work, we assumed that each transaction accesses a set of data items in the same order with (or without) a gap between any two

Received May 28, 2012; Revised January 26, 2013.
The presented work has been conducted and funded by Zarqa University.

data items belonging to the set of all data items (see Definitions 2.5 and 2.6) that have been accessed yet. This paper is organized as follows. In Section 2, we introduce the transactions model and the definitions of serializability, gap, histories and conflict graphs. In Section 3, we prove that the existence of maximum gap (G_{n-2}), in the set of all data items, and the existence of a cycle in the corresponding conflict graph lead to the existence of cycle of length n. Also, the time complexity gained is discussed in Section 3. In Section 4, applications that may satisfy the transactions model are introduced. Conclusion is given in Section 5.

2. Transactions Model and Serializability

In this section we introduce some of the basic definitions of the concurrent transactions, and their histories, which may iterate infinitely many times. In this paper, we assume that the aggregate of all iterations (repetitions) of n transactions will constitute an infinite number of transactions, and consequently infinite histories. But, at any point in time, the number of transactions will be less than or equal n.

2.1. Concurrent Transactions and Histories.

Definition 2.1. *A multi-step transaction T_i, accessing a set of data items $D_i = \{x_1, x_2, \ldots, x_m\}$,* is a sequence (totally ordered set) of read and write steps, as in [12], where every read step $r_i(x)$ precedes a write step $w_i(x)$, $\forall x \in D_i$ such that

$$T_i = r_i(x_1)w_i(x_1)\ldots r_i(x_m)w_i(x_m).$$

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. As in [9], for a history h, we denote $s_i <_h s_{i'}$ if step s_i of T_i occurs before step $s_{i'}$ of $T_{i'}$ in h. In this paper, we shall adopt the serializability as a correctness condition for transactions executing concurrently. The produced history of the transactions should be an 'equivalent' to a serial execution of all the $T_i \in T$. These definitions of equivalence and serializability are actually based on those in [8]. Next, we shall define the serializability and the equivalence between two histories.

Definition 2.2. 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 $i, i' \geq 1, i \neq i'$, and for all $x \in D$,
(1) if $r_i(x) <_{h_1} w_{i'}(x)$, then $r_i(x) <_{h_2} w_{i'}(x)$,
(2) if $w_i(x) <_{h_1} w_{i'}(x)$, then $w_i(x) <_{h_2} w_{i'}(x)$ and
(3) if $w_i(x) <_{h_1} r_{i'}(x)$, then $w_i(x) <_{h_2} r_{i'}(x)$.

Definition 2.3. 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, $\forall i \in \mathbb{N}_1$,

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

such that $h \sim h_S$.

2.2. Conflict Graph and Gap. Usually, the serializability of concurrent transactions can be tested in a polynomial time using *conflict graphs* [7]. The definition below shows how we can build a conflict graph for a history of transactions.

Definition 2.4. *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 *precedence graph* or *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 $T_{i'}$.

Now, two steps are *conflicting* if they belong to different transactions, they access the same data item and at least one of them is a write step. Next, Definitions 2.5, 2.6 and 2.7 give a condition on the set of data items that has been accessed by a set of transactions at any point in time, and generated a history h. This condition will be used to prove that the cycle in the corresponding conflict graph $G(h)$ can be reduced.

Definition 2.5. 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

$$D_i = \{x_{i_1}, x_{i_2}, \ldots, x_{i_l}\},$$

where $x_{i_1} <_D \cdots <_D x_{i_2} \cdots <_D x_{i_l}$ 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 $x_a <_D \ldots <_D x_b$. In Definition 2.5, we have defined formally an order on the set of data items D from which data is accessed by transactions. As the set of data items D is finite, the number of different transactions that can be comprised is also finite. But, the infinite histories are produced when transactions access finite set of data items in a continuous stream.

Definition 2.6. Assume that the transaction T_i accesses a set of data items D_i, where $D_i \subseteq D$ such that

$$D_i = \{x_a, x_b, x_c\}$$

where $x_a <_D \ldots <_D x_b \ldots <_D x_c$. Then the *gap* G^i of the set D_i, will be calculated as follows:

$$G^i = (c - a + 1) - (|D_i|) \qquad (1)$$

where $(c - a + 1)$ is the number of elements in the sequence $x_a \ldots x_c$, and $|D_i|$ is the cardinality of the set D_i.

For example, assume that D_1 and D_2 are

$$\begin{aligned} D_1 &= \{x_3, x_4, x_5\} \\ D_2 &= \{x_1, x_5\}. \end{aligned}$$

Then, by equation (1), $G^1 = 5 - 3 - 3 + 1 = 0$. This means that there is no data items that has not been accessed by T_1 between x_3 and x_5. $G^2 = 5 - 1 - 2 + 1 = 3$. This means that there are 3 data items that have been ignored by T_2 (x_2,x_3 and x_4).

Definition 2.7. Let $T = \{T_i : 1 \leq i \leq n\}$ be a set of transactions that iterates infinitely many times to constitute $T = \{T_i : i \in \mathbb{N}_1\}$ and each $T_i \in T$ accesses a set of data items D_i (as in Definition 2.5). At any point in time there exist k transactions, where $1 \leq k \leq n$, such that

$$\bigcup_{i=1}^{k} D_i = \{x_a, x_{a+1}, \dots, x_u\}$$

where $x_a <_D \cdots <_D x_{a+1} \cdots <_D x_u$. Then, *the maximum gap* G is calculated as follows:

$$G = \begin{cases} 0, & \forall i, 1 \leq i \leq k, G^i = 0 \\ u - a - 1, & \exists i, 1 \leq i \leq k, G^i \neq 0. \end{cases} \tag{2}$$

For example, assume that D_1 and D_2 are

$$\begin{aligned} D_1 &= \{x_1, x_3, x_5\} \\ D_2 &= \{x_3, x_5\}. \end{aligned}$$

Then, by equation (1), $G^1 = 2$ and $G^2 = 1$. Therefore, by equation (2), the maximum gap $G = 5 - 1 - 1 = 3$. Hence, from Definition 2.6 and 2.7, we can say that, as the set of data items D is finite, the maximum gap G will be always finite.

3. Reducing Conflict Graph with Gaps

Lemma 3.1. *If we have a cycle in the conflict graph $G(h)$ of length n, then we have n number of distinct transactions participating in the cycle.*

Proof. Assume that we have a cycle of length n and we don't have n distinct transactions participating in the cycle. This means that the cycle has repeated transactions, as in Figure 1, where the dotted arrow means that there are an unknown number of connected nodes by arcs. Therefore, the first occurrence of T_i (in the cycle) also precedes the transaction T_k in accessing x_i so; there is at least an arc such that (T_i,T_k), see Figure 2. This will reduce the cycle and contradicts with the assumption which says that the cycle is of length n. $\square$

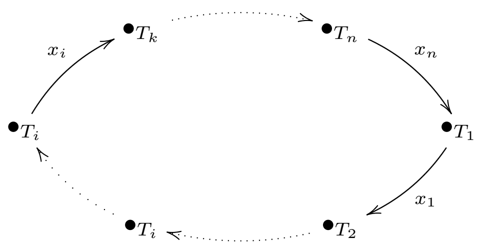


FIGURE 1. Cycle in $G(h)$ of length n.

The next theorem proves that the serializability corresponds to the acyclicity of conflict graph even though the transactions iterate infinitely many times.

Theorem 3.2. *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 (though not necessarily accessing the same data items), is serializable iff the conflict graph $G(h)$ is acyclic.*

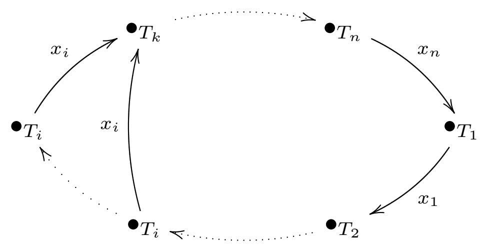


FIGURE 2. Cycle reduction .

Proof. This theorem is proved as Theorem 4 in [10]. □

Theorem 3.3. *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 $n \geq 3$, there are two transactions T_{i1}, T_{i2} such that $G(h)$ has the cycle $(T_{i1}, T_{i2}), (T_{i2}, T_{i1})$.*

Proof. This theorem is proved as Theorem 11 in [9]. □

The above theorem (Theorem 3.3) proves that if there is a cycle in the conflict graph of length n and the set of data items that has been accessed has no gaps (the maximum gap $G = 0$) then, there exists a cycle of length two. Moreover, this theorem will be used in proving Theorem 3.4.

Theorem 3.4. *Let D be a set of data items irreflexively totally ordered such that*

$$D = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$$

and T be a set of transactions that are iterated infinitely many times such that $T = \{T_i : i \in \mathbb{N}_1\}$ and accessing the set D as in Definition 2.7. Assume we have the maximum gap $G = n - 2$, in the set D, denoted by G_{n-2}, and there is a cycle in the corresponding conflict graph $G(h)$. Then, there exists a cycle of length n, denoted by C_n, in the corresponding conflict graph $G(h)$.

Proof. Assume that we have a gap of length $n-2$ (G_{n-2}) and a cycle of length $n+k$ (C_{n+k}), where $k \neq 0$. By Lemma 3.1, we can have a cycle such that:

$$T_1 \overset{x_1}{\to} T_2 \overset{x_2}{\to} T_3 \overset{x_3}{\to} \ldots T_{n+k-1} \overset{x_{n+k-1}}{\to} T_{n+k} \overset{x_{n+k}}{\to} T_1.$$

Therefore, the data items set that have been accessed by the transactions, in that point in time, is

$$\bigcup_{i=1}^{n+k} D_i = \{x_1, \ldots, x_{n+k}\}.$$

So, by equation (2) of Definition 2.7, the maximum gap, $G = n+k-1-1 = n+k-2$ (G_{n+k-2}). Otherwise, by equation (2) of Definition 2.7, the maximum gap $G_{n-2} = 0$. This means that there is no gap. In other words, the transactions, at that point in time, access all data items in D_i ($x_1, \ldots, x_{n+k}$). Hence, by Theorem 3.3, there exists a cycle of length 2. This means that $C_{n+k} = C_2$ ($n+k = 2$) and $G_{n-2} = G_0$ ($n = 2$). Now, its easy to conclude that $k = 0$. This result contradicts with our assumption which says we have $k \neq 0$ and also means we have a cycle of length n (C_n). Moreover, the maximum gap is $G = n + k - 2$ (G_{n+k-2}) and $k \neq 0$. This contradicts our assumption which says that the maximum gap is G_{n-2}. □

For example, assume that we have four transactions T_1, T_2, T_3 and T_4 accessing the set of data items $D = \{x_1, x_2, \dots, x_7\}$, as in Definition 2.5, as follows:

$$\begin{aligned} D_1 &= \{x_3, x_4\}, \qquad D_2 = \{x_3, x_4\}, \\ D_3 &= \{x_3, x_5\}, \qquad D_4 = \{x_4, x_5\}. \end{aligned}$$

According to the Definition 2.6, G^1, G^2, G^3 and G^4 will be 0, 0, 1 and 0, respectively. Also, by equation (2) in Definition 2.7, $G = 1$ (G_1). By Theorem 3.4, the maximum cycle will be of length 3 (C_3). Thus, if we try to build a conflict graph, for the history h of all transactions T_1, T_2, T_3 and T_4, $G(h)$ that contains a cycle, we shall find a cycle of length 3 (C_3). This corresponds to our claim in Theorem 3.4. The main idea behind this paper is to make the test for serializability of transactions computationally efficient. This can be satisfied by testing only the cycle of length $n+2$ if the maximum gap (in the set of data items that has been accessed at that point in time) equals n. In other words, assume that we need to test whether the transactions are serializable (or are in a correct execution) in some point in time. All we need is to test for the existence of a cycle in the conflict graph (at the same point in time) of length $n+2$ if the maximum gap in a set of data items equals n. Moreover, assume that we have 100 transactions, at any point in time, contending to access a set of data items D, where $|D| = 50$ and the maximum gap $G = 6$. Now, we only need to test a cycle of length $n = 8$ rather than cycle of length 100. This result is very useful for reducing the time complexity of checking the correctness of the transactions. The next subsection discusses the time complexity of checking the serializability of such transactions.

3.1. Time Complexity.

Assume that we have n transactions (in a database scheduler) at a point in time, and we need to test whether the history that contains the transactions is serializable or not. Now, to check for the existence of a cycle of length k, the time complexity will be as follows:

$$\begin{aligned} TC_k(n) &= n(n-1)...(n-k+1) \\ &= \prod_{i=0}^{k-1}(n-i) \end{aligned} \tag{3}$$

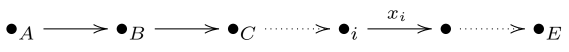
Now, to test for the existence of a cycle of any length, we test for the existence of cycles of length $2, 3, ...n$ as follows

$$\begin{aligned} TC(n) &= \sum_{i=2}^{n} TC_i(n) \\ &= \sum_{i=2}^{n}\prod_{j=0}^{i-1}(n-j) \end{aligned} \tag{4}$$

where $TC(n)$ denotes the time complexity of testing for a cycle of any length. Now, from equation (4), it is easy to show that the time complexity to test a cycle of any length $TC(n) \in O(n!)$. But, to test for a cycle of length k (if the maximum gap $G = k - 2$), see equation (3), the complexity is $TC_k(n) \in O(n^k)$. Clearly, the performance of testing for a cycle of k transactions is much better than testing all cycles. Moreover, if the maximum gap $G = n - 2$ (therefore, the length of cycle equals n (C_n)) and the number of concurrent transactions at that point in time equals k such that $k < n$, then we only need to test for a cycle of length k. This result can be concluded from Lemma 3.1.

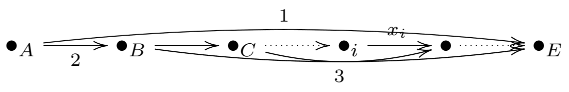
4. Applications

It becomes widely common to book an e-ticket from booking agencies around the world. Clearly, the list of destinations is naturally ordered somehow. Therefore, a passenger intending to book a ticket from location A to E has many choices depending on the availability of destinations. To demonstrate this situation, we assume that the set of all available destinations D (where $|D| = k$) are ordered as in Figure 3. Also, the set D represents the available destinations starting from location A and ending at location E. x_i represents the next destination from location i such that $x_i \in D$, see Figure 3. The first choice, for the passenger, is that he/she can book a ticket directly from A to E without transiting via any destination between A and E. The transaction T_1, that represents this choice, accesses the set $D_1 = \{A, E\}$. From Definition 2.6, the gap, of the set D_1 is $G^1 = k - 2$. This choice is depicted as arc 1 in Figure 4. The second choice is that he/she can book a ticket from A to B and then from B to E, see arcs 2 and 3 in Figure 4, and so on. This scenario can be implemented as a multi-step transaction accessing ordered data with the existence of gaps, where a read step corresponds to browsing journey times from a destination, and the write step represents the booking of a chosen time to the next destination in the order, see Definition 2.1. The set of ordered destinations represents the ordered set D that has been defined in Definition 2.5. The gap represents the number of destinations that have been ignored in transiting, between A to E, see Definition 2.6. Finally, the maximum gap represents the number of all destinations that have been ignored in transiting, if there is an available itinerary from A to E. The increase of number of people using internet and also the increase in the number of passengers around the world make the number of e-tickets transactions, incoming to and outgoing from a web server, unknown. Even though, at any point in time, the number of active transactions in a server is finite.



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

Figure 3. Ordered set of destinations.



$$\begin{tikzcd}
\bullet_A \arrow[r, "2"'] \arrow[rrrrr, bend left, "1"] & \bullet_B \arrow[r] \arrow[rrrr, bend right, "3"'] & \bullet_C \arrow[r, dotted] & \bullet_i \arrow[r, "x_i"] & \bullet \arrow[r, dotted] & \bullet_E
\end{tikzcd}$$

Figure 4. Ordered set of destinations with existence of gaps.

5. Conclusion

The traditional database concurrency control techniques deal with a finite number of transactions that are executing concurrently in a system. But, the development in wireless communications (or mobile computing) and portable devices has led to a huge number of users executing their transactions concurrently at the same time.

In this paper, we have proven that the conflict graph can be reduced, under certain condition on the set of data items. This reduction makes the time complexity for testing the serializability of transactions much better than testing the serializability of all transactions participating in the conflict graph. Moreover, this reduction can be used to specify the correctness of infinite histories that can model such large numbers of transactions in the case where the data accessed are ordered. The main advantage of this result, that has been given, is that the testing for serializability only requires considering k number of transactions, if the maximum gap equals $k - 2$. This makes testing for serializability efficient and easy to encode into the widely used temporal logics CTL and LTL.

The modelling and verification of a scheduler include representing all possible histories of transactions as a finite state transition system. The serializability of k transactions, given here, are encoding into temporal logics (either CTL and LTL) formula ϕ. Then, the model checker automatically runs to ensure that all possible histories satisfy the formula ϕ. This can all be done using common model checkers such as NuSMV and SPIN.

Further work will attempt 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 such as debit and credit transactions.

Acknowledgement

We would like to thank ZU (Zarqa University) for its support and grant in making this work possible.

References

[1] A. Cimatti, E. Clarke, F. Giunchiglia and M. Roveri, *NuSMV: a new symbolic model verifier*, Lecture Notes in Computer Science 1633 (1999), 495–499.

[2] E. Clarke, O. Grumberg and D. Peled, *Model checking*, MIT Press, 1999.

[3] M.P. Fle and G. Roucairol, On serializability of iterated transactions, *Proc, 1st ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 1982, 194–200.

[4] W. Hussak, Serializable histories in Quantified Propositional Temporal Logic, *International Journal of Computer Mathematics* **81** (2004), no. 10, 1203–1211.

[5] W. Hussak, Specifying strict serializability of iterated transactions in Propositional Temporal Logic, *International Journal of Computer Science* **2** (2007), no. 2, 150–156 (at `www.waset.org/ijcs`).

[6] W.Hussak, The serializability problem for a temporal logic of transaction queries, *Journal of Applied Non-Classical Logics* **18** (2008), no. 1, 67–78.

[7] C.H. Papadimitriou, The serializability of concurrent database updates, *Journal of the ACM* **26** (1979), 631–653.

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

[9] R. Alshorman and W. Hussak, A Serializability Condition for Multi-step Transactions Accessing Ordered Data, *International Journal of Computer Science* **4** (2009), no. 1, 13–20.

[10] R. Alshorman and W. Hussak, A CTL Specification of Serializability for Transactions Accessing Uniform Data, *International Journal of Computer Science and Engineering* **3** (2009), no. 1, 26–32.

[11] G.J. Holzmann, *The SPIN model checker*, Addison-Wesley, 2004.

[12] N. Katoh, T. Kameda and T. Ibaraki, A cautious scheduler for multi-step transactions, *Algorithmica* **2** (1987), no. 1–4, 1–26.

(Rafat Alshorman, Hamed Fawareh) Department of Computer Science, Zarqa University, Zarqa, Jordan
E-mail address: `rafat_sh@zu.edu.jo, fawareh@zpu.edu.jo`