Globally Consistent Event Ordering In One-Directional Distributed Environments

Paul Ammann [*] Sushil Jajodia [†]
Center For Secure Information Systems
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030

Phyllis Frankl [‡]
Computer Science Department
Polytechnic University, Brooklyn, NY 11201

Technical Report ISSE-TR-93-104
August, 1993

[*]Supported in part by the National Science Foundation under grant CCR-9202270.
[†]Supported in part by a DARPA grant administered by the Office of Naval Research under grant N0014-92-J-4038.
[‡]Supported in part by NSF Grant CCR-9206910 and by the New York State Science and Technology Foundation Center for Advanced Technology program.

Abstract

We consider communication structures for event ordering algorithms in distributed environments where information flows only in one direction. Example applications are multilevel security and hierarchical databases. Although the most general one-directional communication structure is a partial order, partial orders do not enjoy the property of being *consistently-ordered*, a formalisation of the notion that globally consistent event ordering is ensured. Our main result is that the *crown-free* property is necessary and sufficient for a communication structure to be consistently-ordered. We discuss the computational complexity of detecting crowns and sketch typical applications.

Index Terms

Crown-free partial orders, Distributed networks, Event ordering, Graph algorithms, Hierarchical databases, Lamport clocks, Multilevel databases, Security and protection.

Address for Correspondence

Point of Contact	Paul Ammann
Address	Department of Information and Software Systems Engineering
	S&T II, Room 430
	George Mason University, Fairfax, VA 22030
Email	pammann@isse.gmu.edu
Office	(703) 993-1660
Dept. Secretary	(703) 993-1640

1 Introduction

We consider the ordering of events in a distributed environment. For our purposes, a distributed environment is one where a set of events compose some computation, the events occur at a network of sites, and sites communicate by passing messages. Some applications impose restrictions on the communication structure, and these restrictions can be exploited to make more efficient ordering decisions. In this paper we consider restrictions that force communication to occur in one direction only. Our particular focus is to determine the largest class of such communication structures that are capable of guaranteeing consistent ordering decisions without resort to further global synchronization.

A common task in a distributed environment is to decide the ordering of any pair of events. A standard approach to this problem is Lamport's timestamp algorithm [Lam78], which can be used to impose a partial order on events. The basic idea behind Lamport's algorithm is that each event at a site is marked with a unique local timestamp. Timestamps are drawn from some monotonically increasing sequence, such as an integer counter. If site A sends a message to site B, then A includes the most recent timestamp at A in the message. Upon receipt of the message, B increases its current timestamp to the timestamp value in the message, if necessary. The ordering of any pair of events is determined in part by consulting the corresponding timestamps and in part by consulting the record of message receipts.

Given events e_1 and e_2, Lamport's algorithm yields three possible outcomes, namely that e_1 precedes e_2, that e_2 precedes e_1, or that e_1 and e_2 are concurrent. The interpretation of the last possibility is that it is unknown, and perhaps unimportant, which of e_1 or e_2 'really' happened first. If all events must be ordered, concurrent events can be forced into a partial order according to the corresponding timestamps. Resolving the case of two identical timestamp values with a static precedence order on sites yields a total order.

A motivation for this paper is the observation that some of the order-

ing decisions made to obtain a total order are inherently artificial, and so, unsurprisingly, no single algorithm suits all applications. For example, in a transaction processing application, suppose that a site receives a message about a transaction that a timestamping algorithm determines to be far in the past at the receiving site. If the site is to accept the message, then the site may be obliged to unwind all later transactions, incorporate the effect of the remote transaction, and then redo the unwound transactions. From the perspective of the receiving site, it would be more efficient for the ordering of the event to be determined, at least in part, by the clock at the site that receives the message rather than by the clock at the site where the event occurred.

Allowing event ordering to be determined upon receipt of a corresponding message at a given site can lead to inconsistencies. For example, suppose that event e_1 occurs at site A and, concurrently, event e_2 occurs at site B. Let A send a message about e_1 to B and B send a message about e_2 to A. The ordering choices that involve the least rework are for A to order e_1 prior to e_2 and for B to order e_2 prior to e_1. Each local order is consistent, but there is no global consistent order.

This simple example shows that if ordering of otherwise concurrent events is to be done by the site receiving a message, the structure by which sites communicate must be antisymmetric. Otherwise, in the absence of some other global synchronization mechanism, the type of inconsistency exhibited above can arise. It is not difficult to extend the above example to show that the transitive closure of the communication structure must also be antisymmetric, and hence an acyclic communication structure is necessary in general.

Although a restriction to acyclic communication structures might seem too strong to be useful, there are applications, such as multilevel security [BL76, Den82] and hierarchical databases [HC86], that mandate such restrictions on information flow. For example, in a multilevel security environment, suppose that site A encapsulates a database at the 'Secret' level and site B encapsulates a database at the 'Unclassified' level. Communication from A

to B is prohibited to prevent the leakage of classified information. It is satisfactory for A to make an ordering decision about an event at B upon receipt of the corresponding message since B has no opportunity to make a conflicting decision. Indeed, the existence of events at A, as well as any information that depends on events at A, must be hidden from B to satisfy the security requirements.

A general acyclic structure is a partial order, but it is known that partial orders by themselves do not lead to global consistent orderings, as is reviewed by example later in the paper (*c.f.* fig. 1). In the multilevel security domain, a restriction on partial orders to lattices is common [BL76, Den82]. A lattice is a partial order in which each pair of sites has a unique least upper bound and and a unique greatest lower bound. Unfortunately, lattices by themselves also do not lead to consistent orderings. In [AJ93b], it was shown that for a variety of concurrency control algorithms for multilevel, replicated, secure databases [JK90, Cos92, KK92], lattices do permit inconsistent orderings, *i.e.* unserializable execution histories. Serializability is a major concern in database concurrency control, and conditions that lead to serialization problems are correspondingly important. A restriction to planar lattices is sufficient to avoid the identified serializability problem for the cited concurrency control algorithms, but planarity is not a necessary condition. In a related paper, it was shown that for component-based timestamp generation [AJ93a], a planar lattice is sufficient, but again unnecessary, for consistent ordering.

In this paper, we develop a formal notion called the *consistently-ordered* property to describe communication structures that ensure globally consistent ordering decisions without resort to further global synchronization. Our main result is showing that the consistently-ordered property is equivalent to a standard characterization of partial orders known as the *crown-free* property. In terms of the previous paragraph, we show that crown-freedom is both a necessary and a sufficient condition to guarantee the consistently-ordered property. The absence of crowns in partial orders has other useful applications, for example [DRW82] exploit crown-free partial orders to develop an

efficient scheduling algorithm.

The structure for this paper is as follows. In section 2, we supply a model for event ordering in a distributed environment with a one-way communication structure. The model yields a formal definition of the consistently-ordered property. In section 3, we introduce the crown-free property and show that it is equivalent to the consistently-ordered property. In section 4, we discuss the computational complexity of deciding whether a communication structure is crown-free. In section 5 we make some observations about our results and sketch applications to multilevel security and hierarchical databases. The reader unfamiliar with these applications may wish to read section 5.2 first. In section 6 we conclude the paper.

2 Definitions

We begin with a remark on the notation. We have chosen to present our formalisms with the conventions of the **Z** notation. For the most part, **Z** notation follows typical set theory; where different, we give an explanatory note.

Let $Classes$ be a set, an element of which we refer to as a *class*. As an example, in a multilevel security application $Classes$ might correspond to all possible security classifications, such as 'Confidential', 'Secret-NATO', and so on. Let $\mathbf{P} = \{P_1, P_2, ...\}$ be some finite subset of $Classes$, and let $<$ be a relation on $\mathbf{P}$ that is antisymmetric and transitive. To extend the multilevel security example above, $\mathbf{P}$ corresponds to those security classifications that are actually employed, and $<$ is the dominance relation between security classifications.

$\mathbf{S} = (\mathbf{P}, <)$ is a *partial order*. If $P_j < P_i$, we say that P_i dominates P_j, which we also write $P_i > P_j$. We write $P_j \leq P_i$, to allow the case where $i = j$, and $P_j <> P_i$ to denote that P_i and P_j are incomparable. Finally, we assume that $\mathbf{S}$ has a greatest element; we discuss relaxing this assumption later in the paper.

Let $Events$ be a set, an element of which we refer to as an *event*. As

an example, in a database application $Events$ might correspond to the set of all possible transactions. Let $\mathbf{E} = \{e_1, e_2, ...\}$ be some finite subset of $Events$. To extend the database example, $\mathbf{E}$ typically corresponds to the set of committed transactions.

We associate every event in $\mathbf{E}$ with some class in $\mathbf{P}$. The (partial) function $L : Events \rightarrow Classes$ captures this mapping. Combining our prior examples, we might associate each transaction with a particular security classification, *e.g.* $L(e_1) = Secret$ means that transaction e_1 is classified as 'Secret'.

Event e is said to be *local* to P if $L(e) = P$. An event e is *visible* at class P if there exists a $P_j \leq P$ such that e is local to P_j. We define $\mathbf{E}_P$ to be the set of all events local to P: formally $\mathbf{E}_P = \{e : Events \mid L(e) = P\}$. Informally, this definition reads, '$\mathbf{E}_P$ is the set of all e of type $Events$ such that $L(e)$ is P'.

We require that local events at any given class be totally ordered. At first, it might appear that it would be more desirable to only require a partial order on local events. However, such an approach allows remote sites to extend the partial order inconsistently. For example, in a transaction processing context, our model requires each local site to produce a total serialization order for local transactions, even though some pairs of transactions might not conflict. The reason is that two remote sites might induce different serialization orders by the scheduling of further transactions, thus precluding a global consistent order.

To model the total order of local events, let $\mathbf{A}_P$ be a sequence of the events in $\mathbf{E}_P$. $\mathbf{A}_P$ is total and injective with respect to $\mathbf{E}_P$; each local event in $\mathbf{E}_P$ appears exactly once in $\mathbf{A}_P$.

In our model, ordering decisions at class P are constrained only by ordering decisions at dominated classes, *i.e.* classes P_j, where $P_j \leq P$. Suppose e_1 and e_2 are events visible at P. If e_1 and e_2 are ordered at some class P_j where $P_j < P$, then class P must respect that ordering. On the other hand, if e_1 and e_2 are not ordered at any class P_j where $P_j < P$, then class P is free to choose either ordering for the two events. Also, for the same reason that

each site must totally order local events, each site must also totally order all visible events. We introduce some machinery to formally express this idea.

We define the *subsequence* predicate, whose signature is:

$$\_ \sqsubseteq \_ : seq[Events] \times \ seq[Events] \rightarrow Boolean$$

as follows. Let A and B be two sequences of events. Then $A \ \sqsubseteq \ B$ is true iff A can be obtained from B by discarding events in B.

We define $global(P)$ to be an injective sequence of events visible at P such that:

1. $\mathbf{A}_P \sqsubseteq global(P)$
2. $\forall P_j : \ Classes \mid P_j < P \bullet global(P_j) \sqsubseteq global(P)$

The first condition on $global(P)$ states that $global(P)$ must respect the local order at P. The second condition on $global(P)$ states that $global(P)$ must respect the total orders chosen at all dominated classes P_j. The second constraint reads, 'for all P_j of type $Classes$ where P_j is dominated by P, it is the case that $global(P_j)$ is a subsequence of $global(P)$'. Note that P is a free variable in both constraints.

For a given partial order $\mathbf{S}$ and set of events $\mathbf{E}$, there may be many possible values $global(P)$, *i.e.* *global* is a relation, not a function. However, for some choices of $global(P_j)$ at $P_j < P$, $global(P)$ may not exist. This possibility is exhibited in table 1 and figure 1. Suppose $\mathbf{E} = \{e_1, e_2\}$ and

$class$	$\mathbf{E}_{class}$	$\mathbf{A}_{class}$	$global(P_{class})$
1	$\{e_1\}$	$\langle e_1 \rangle$	$\langle e_1 \rangle$
2	$\{e_2\}$	$\langle e_2 \rangle$	$\langle e_2 \rangle$
3	$\emptyset$	$\langle\rangle$	$\langle e_1, e_2 \rangle$
4	$\emptyset$	$\langle\rangle$	$\langle e_2, e_1 \rangle$
max	$\emptyset$	$\langle\rangle$	does not exist

Table 1: Event And Ordering Assignments

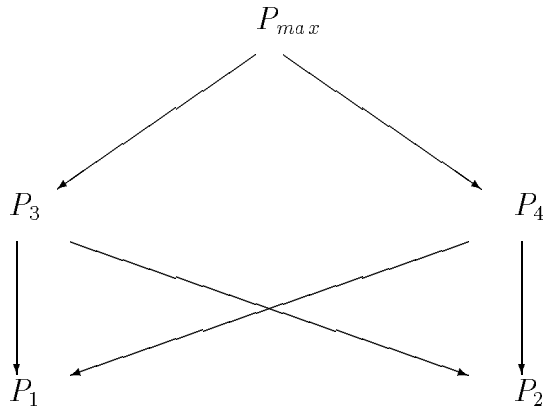

Figure 1: Example Partial Order That Is Not Consistently-Ordered

consider the event and ordering assignments given in table 1. For the given events and orderings, $global(P_{max})$ does not exist.

To provide a more concrete interpretation of the difficulty exhibited in table 1, suppose that e_1 and e_2 are transactions at classes P_1 and P_2, respectively. Further suppose that $global(P)$ represents some equivalent serial order for the execution history of all transactions visible at P. In the example shown, the value of $global(P_3)$ indicates that P_3 serializes e_1 before e_2, perhaps by the scheduling of some (unshown) local transaction. Similarly, the value of $global(P_4)$ indicates that P_4 serializes e_2 before e_1. Since $P_3 <> P_4$, P_3 and P_4 do not communicate and are unable to detect that a global serialization anomaly has arisen. However, the inconsistent serialization is apparent at P_{max}, and is reflected by the fact that $global(P_{max})$ does not exist.

The purpose of this paper is to find the largest class of partial orders that still guarantees the existence of of $global(P)$ for any class P, no matter what choices are made in constructing $global(P_j)$ for dominated classes $P_j < P$. Informally, each class in a conforming structure can make arbitrary ordering decisions about otherwise unordered events and still be guaranteed that the

resultant global ordering is consistent. We formalize the notion of consistent global ordering as follows:

Definition: The partial order **S** is *consistently-ordered* at P iff for all classes $P_j < P$ and for all possible sequences of events $global(P_j)$, there exists at least one $global(P)$. To extend the notion of being consistently-ordered to the entire partial order, we say that **S** is *consistently-ordered* iff **S** is consistently-ordered at the greatest element.

As an example, the partial order in figure 1 is consistently-ordered at P_i for $i \in 1, \ldots, 4$, but not at P_{max}, as shown by the choices exhibited in table 1. Hence the partial order in figure 1 is not consistently-ordered.

3 The Crown-Free Property

Let $D = (V_D, E_D)$ be a graph. $Q = (V_Q, E_Q)$ is an *induced subgraph* of D if $V_Q \subseteq V_D$ and $E_Q = \{e : E_D | \text{the head and tail of } e \text{ belong to } V_Q\}$. Informally, an induced subgraph retains as many edges as possible from the parent graph.

A *crown* in $\mathbf{S} = (\mathbf{P}, <)$ is a subset $\{x_1, \ldots, x_{2n}\}$ of **P** such that $x_i < x_j$ iff $i \in 1, \ldots, n, j \in n+1, \ldots, 2n$, and $i = (j-1 \ mod \ n)+1$ or $i = (j \ mod \ n)+1$. A crown is exhibited in figure 2. (With some imagination, the 'crown' can be envisioned in three dimensions.) S is *crown-free* iff S has no crown. The above definition of a crown is a variation of one in [Bou85] and is similar to [Riv85, page 531].

Crowns in directed graphs are defined analogously to crowns in partial orders. Note that a directed acyclic graph D contains a crown iff there exists an induced subgraph Q of D such that:

1. Q is bipartite.

2. The undirected version of Q is cyclic.

For example, the directed graph shown in figure 1 has a crown since the desired Q can be obtained by discarding P_{max}. To continue the example, let

$\mathbf{S} = (\mathbf{P}, <)$ be the partial order whose Hasse diagram is shown in figure 1 and D be the directed graph whose nodes are $\mathbf{P}$ and whose edges are $<$. Note that, by definition, $<$ is transitively closed even though the corresponding Hasse diagram, *e.g.* figure 1, does not show the transitive edges for purposes of clarity. S has a crown since if we discard P_{max} from D, we obtain the same Q as before.

We now give the main result of the paper:

Theorem 1 $\mathbf{S}$ *is consistently-ordered iff* $\mathbf{S}$ *is crown-free.*

Proof:

$\rightarrow$: Suppose $\mathbf{S}$ is consistently-ordered. Then $\mathbf{S}$ is crown-free.

For the sake of contradiction, consider an $\mathbf{S}$ that has at least one crown. Let Q be such a crown. Label the minimal n nodes in Q as $P_1, \ldots, P_n$ and label the maximal n nodes in Q as $P_{n+1}, \ldots, P_{2n}$. Define $\mathbf{E}_i = \{e_i\}$ for $i \in 1, \ldots, n$; thus $\mathbf{A}_i = \langle e_i \rangle$ for $i \in 1, \ldots, n$. Without loss of generality, relabel the $2n$ nodes in Q as necessary such that P_i, $i \in n+1, \ldots, 2n$, dominates $P_{(i-1 \ mod \ n)+1}$ and $P_{(i \ mod \ n)+1}$, as in figure 2. Let $global(P_i)$, $i \in n+1, \ldots, 2n$, be $\langle e_{(i-1 \ mod \ n)+1}, \ e_{(i \ mod \ n)+1} \rangle$. Let P_{max} denote the maximal element in $\mathbf{P}$. Then $global(P_{max})$ does not exist, since we are requiring the "sequence" $global(P_{max})$ to accommodate the orderings:

$\langle e_1, e_2 \rangle$
$\langle e_2, e_3 \rangle$
...
$\langle e_n, e_1 \rangle$

which is impossible. Hence $\mathbf{S}$ is not consistently-ordered, which is a contradiction. Therefore, $\mathbf{S}$ is crown-free.

□

$\leftarrow$: Suppose $\mathbf{S}$ is crown-free. Then $\mathbf{S}$ is consistently-ordered.

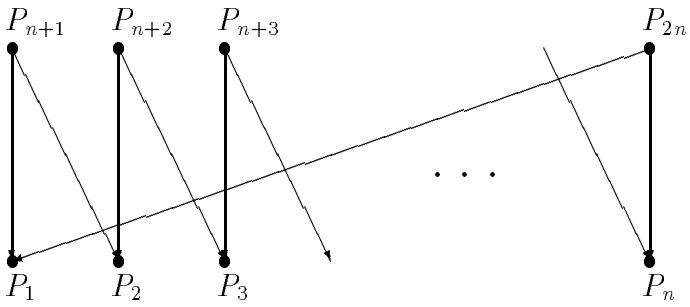

Figure 2: A Crown

For the sake of contradiction, consider a partial order **S** that is not consistently-ordered. Consider P, a minimal element in **P** such that $global(P)$ does not necessarily exist.

Consider an instance where P is unable to form $global(P)$. It must be the case that P is faced with an inconsistent set of event ordering requirements. Each event ordering is of the form $\langle e_x, e_y \rangle$, and each requirement to so order the events is imposed by some $global(P_j)$, where $P_j < P$.

Suppose that T is a minimal sized set of inconsistent event orderings. Let the size of T be n, and relabel the events as needed such that T may be listed as:

$\langle e_1, e_2 \rangle$
$\langle e_2, e_3 \rangle$
...
$\langle e_n, e_1 \rangle$

We make several observations about T. Since T is assumed to be of minimal size, each event in the list appears exactly twice. Also, for each event e_i, $L(e_i) < P$.

Since T is minimal, we have that $L(e_i) <> L(e_j)$ unless $i = j$. To see why, suppose that $L(e_i) < L(e_j)$. Then the ordering of e_i and e_j is done

exactly once done, namely at $L(e_j)$. Hence by substituting all occurrences of e_j for e_i we could reduce the size of T, which is a contradiction. Similar arguments apply if $L(e_i) = L(e_j)$ or $L(e_i) > L(e_j)$.

Thus there are n incomparable classes where the n events giving rise to T are local. We collect these n classes in the set B_{low}. Note that B_{low} is an antichain.

For each $\langle e_i, e_{(i \bmod n)+1} \rangle$ in T, consider the least upper bound class of $L(e_i)$ and $L(e_{(i \bmod n)+1})$. The claim is that there are exactly n distinct least upper bound classes for the pairs of classes in T. To see why, suppose that there there are more than n least upper bound classes. Then some pair of classes in T would have at least two least upper bounds, and from figure 1 it is clear that **S** has a crown, which is a contradiction. Now suppose that there are fewer than n least upper bound classes. Then at least two pairs in T must share a least upper bound, denoted P_j. These two pairs must comprise at least 3 distinct events, and $global(P_j)$ totally orders these events. But then T is not of minimal size, which is a contradiction.

We collect the n least upper bound classes into the set B_{high}. Classes in B_{high} are incomparable, or else we could again argue that T is not minimal. Note that B_{high} is also an antichain.

Now, let Q be a directed graph whose nodes are $B_{low} \cup B_{high}$ and whose edges defined by $<$. Q is bipartite and its undirected version is cyclic. The cycle is exhibited by the structure T. Therefore **S** has a crown, namely Q, which is a contradiction. Again, figure 2 provides an illustration. □

4 Computational Complexity

We now use the results of section 3 to give a polynomial-time algorithm for determining whether a communication structure **S** ensures globally consistent event orderings. By Theorem 1, it suffices to show that **S** is crown-free. The naive algorithm, explicit checking of each subset for the bipartite property and for the existence of a cycle, is obviously exponential. Bouchitté [Bou85] gives a polynomial-time algorithm, which we summarize below, for the crown

detection problem. The crown detection algorithm is based on deriving a bipartite graph from the partial order, then using an 'elimination scheme' to examine this bipartite graph.

The *split graph* of a partial order $(\mathbf{P}, <)$ is the bipartite graph $G = (V, V', E)$ where each $x \in \mathbf{P}$ is associated with one vertex $v \in V$ and one vertex $v' \in V'$; there is an edge (v, w') in E if and only if $x < y$ where v is associated with x and w' is associated with y. Bouchitté [Bou85] and Trotter [Tro81] establish that crowns in $(\mathbf{P}, <)$ give rise to crowns in the split graph and that any crown in the split graph having more than 4 nodes comes from a crown in $(\mathbf{P}, <)$. Thus, it suffices to check $(\mathbf{P}, <)$ for crowns of size exactly four and then check the split graph, which is bipartite, for the existence of crowns.

Note that a bipartite graph has a crown of size greater than 4 if and only if it has a chordless cycle of length greater than 4. A bipartite graph in which every cycle of length greater than 4 has a chord is called *chordal*. Bouchitté shows that bipartite graph G is chordal if and only if the following iterative procedure results in a graph with no edges:

```
    while not done do
(1)       if G contains a vertex with only one neighbor,
             then remove such a vertex (and the incident edge)
(2)       else if G contains an edge (x,y) such that every
             neighbor of x is adjacent to each neighbor of y
             and vice-versa then remove such an edge (i.e.,
             remove the two vertices and all incident edges).
(3)       else done:= true
    if G has no edges
          then return (CHORDAL)
          else return (NOT CHORDAL)
```

Let G and G' denote the graph at the beginning and end, respectively, of an iteration. The correctness of the algorithm follows from the facts that

Every chordal graph has an edge of the type described in condition (2) [GG78]

G is chordal iff G' is chordal [Bou85]

No edge can be eliminated from a chordless cycle of length greater than 4 [Bou85, GG78]

Let n be the number of nodes and m the number of edges in the graph arising from $(\mathbf{P}, <)$. Checking for crowns of size 4 can be done by brute force in time $O(n^4)$. The split graph has $2n$ nodes and m edges and can be constructed in time $O(m)$. The elimination algorithm has at most $2n + m$ iterations. In the worst case, each iteration involves time $O(n)$ to search for a vertex that can be eliminated plus $O(m)$ checks for whether an edge can be eliminated, each of which takes time $O(n^2)$. Thus, the total running time is $O(m^2n^2) = O(n^6)$. In practice, it should be possible to substantially improve the running time by precomputing a list of candidates for elimination, then updating this list on each iteration of the loop.

We remark that the result of major importance is that the complexity of crown detection in partial orders is polynomial rather than NP-complete. We do not address the complexity of crown detection in arbitrary directed graphs since the issue is not directly relevant to this paper. However, given the variance in complexity results for standard graph problems dependent on whether the graph in question corresponds to a partial order [Moh], it is possible that crown detection in arbitrary directed graphs is NP-complete.

5 Discussion

In this section, we make some observations on our results and discuss how our results can be applied to problems in two areas, namely multilevel security and hierarchical databases.

5.1 Observations

For the analysis given so far, we have assumed that the partial order $\mathbf{S} = (\mathbf{P}, <)$ has a greatest element. (Recall that a partial order has a greatest element iff the partial order has a unique maximal element). The reason for the assumption is as follows. Suppose $\mathbf{S}$ has no greatest element, that $\mathbf{S}$ has at least one crown, and that for every crown Q in $\mathbf{S}$, at least one node in Q is a maximal element in $\mathbf{S}$. In this scenario, there is clearly still the possibility for globally inconsistent event ordering, but no node in $\mathbf{S}$ is in a position to observe the inconsistency. This lack of an observer complicates the discussion, and so, for our analysis, we assumed that $\mathbf{S}$ had a greatest element.

For practical purposes, it is not that important whether S has a greatest element or not. In either case, if $\mathbf{S}$ has a crown, then globally inconsistent ordering decisions are possible, if not necessarily observable.

Our second set of observations relates the crown-free property to some other typical classifications of partial orders. An often employed special case of a partial order is a *lattice*. In a lattice, each pair of classes has a unique least upper bound and a unique greatest lower bound. Lattices are not necessarily crown-free, as can be seen by considering the subset lattice on a set with three elements. (See [AJ93a] or [AJ93b] for an elaboration of this example).

Another intuitively appealing special case of a partial order is a *planar* partial order. A partial order is planar if its Hasse diagram is planar and each edge in the Hasse diagram is monotone, *i.e.* edges are prohibited from 'looping around' the outside of the diagram. (See [Riv85, The Diagram] for a fuller explanation.) By consulting the results of [Riv85], where the types of structures that all *nonplanar* partial orders must contain are enumerated, we note that the structure Q in figure 2 is on the proscribed list [Riv85, figure 18, page 121]. Thus we can be sure that all planar partial orders are crown-free. One can also see directly that the structure in figure 2 is nonplanar. The existence of other structures on the list in [Riv85] demonstrates that although the restriction to planarity is a sufficient condition for a partial

order to be consistently-ordered, it is not a necessary condition.

5.2 Applications

A primary concern in multilevel security is information leakage, such as information in a 'Secret' database leaking to some process executing at the 'Unclassified' level. Leakage can occur in two ways - directly through an overt operation such as reading a data item or indirectly through a covert or signaling channel. Direct leakage can be accounted for by following so-called mandatory access control policies such as the Bell-Lapadula model [BL76, Den82].

Indirect leakage is more troublesome. In a covert or signaling channel, information leaks by means of contention over some resource [BL76, Den82, Lam73]. An example channel is provided by the read and write locks in a conventional database. In a database with a locking protocol for concurrency control, a read (or write) of a data item must be preceded by the acquisition of a read (or write) lock. A request for a write lock, potentially made by a transaction at the 'Unclassified' level, is delayed if a read lock has already been granted, perhaps to a transaction at the 'Secret' level. The delay experienced by the 'Unclassified' transaction can be used to infer activity at the 'Secret' level. Hence 'Secret' information leaks to the 'Unclassified' level if 'Secret' transactions can obtain standard read locks on 'Unclassified' data items.

The problem of covert or signaling channels has been extensively studied. One general approach is to physically separate components at one security level from components at another, thus simplifying the argument that indirect channels do not arise. A natural outgrowth of this trend is a distributed implementation with the type of one-directional communication structure that is the subject of this paper. In particular, multilevel, replicated secure databases, first identified in [Com83], lend themselves to distributed implementations.

Candidate concurrency control algorithms for multilevel, replicated se-

cure databases appear in [JK90, Cos92, KK92]. For the most part, these algorithms assume that the communication structure between different security classes is a lattice for reasons outlined in [Den82]. However, as noted above, some lattices have crowns, and hence without additional synchronization information, distributed implementations of multilevel, replicated databases cannot guarantee serializable execution transaction histories for lattices. Recognition of this problem in the published concurrency control algorithms was in fact the beginning of the present paper. Given the interest in constructing multilevel systems, a precise demarcation of the problematic structures was clearly required. Solutions to the dilemma are to ensure that the communication structure is crown-free, to modify the structure to be crown-free if it is not, or to add additional synchronization measures, such as a global clock.

Communication structures similar to those in multilevel security appear in hierarchical databases [HC86]. Hierarchical databases partition a global database by the access characteristic of transactions. A typical hierarchical database might have a main database containing 'raw' information, and derivative databases where transactions read, but do not write the raw data. The reader is referred to [HC86] and [AJ93a] for more explanation of hierarchical databases.

Hierarchical databases have no information leakage requirements. However, it is still undesirable for a derivative database to interfere with the concurrency control at a main database. Such interference could take the form of holding read locks on raw data items or forcing transactions at a main database to abort to preserve serializability in a timestamp-ordering protocol. The results of this paper can be applied to hierarchical databases as follows: if the communication structure of a hierarchical database is restricted to be crown-free, then a distributed implementation that guarantees globally serializable transaction histories is possible without the introduction of additional synchronization information.

6 Conclusion

Networks in which information flow is restricted to one direction in a distributed network figure prominently in applications such as multilevel security and hierarchical databases. In such networks, the requirements for event ordering differ substantially from unrestricted networks, and indeed improvements in ordering decisions are possible.

In this paper, we have defined the *consistently-ordered* property to describe communication structures where local ordering decisions are guaranteed to be globally consistent without the introduction of additional synchronization, such as a centralized clock. For our main result, we employed the *crown-free* property of partial orders to prove that a crown-free partial order is equivalent to a consistently-ordered one. Fortunately, crown detection can be carried out in polynomial time.

The results in this paper can be applied in any application with one-directional communication structures, such as multilevel security or hierarchical databases, to ensure that distributed applications enjoy desirable properties, such as serializable execution histories.

Future Work

In [AJ93a], a component-based, timestamping algorithm was developed for planar lattices. The results of this paper indicate that the timestamping algorithm applies to crown-free partial orders. In [HC86], a proof technique called the *partitioned synchronization rule* was presented for demonstrating the correctness of database concurrency control algorithms. The proof technique was proven for communication structures that are restricted to semitrees. The results of this paper indicate that the partitioned-synchronization rule applies to crown-free partial orders. Items for future work are to verify these two conjectures.

Acknowledgements

It is a pleasure to acknowledge Ivan Rival for sharing his expertise on partial orders, John McDermott for discussing the problem of consistent ordering, and Jeff Salowe for considering the computational aspects of crown detection.

References

[AJ93a] Paul Ammann and Sushil Jajodia. Distributed timestamp generation in planar lattice networks. *ACM Transactions on Computer Systems*, August 1993. To appear.

[AJ93b] Paul Ammann and Sushil Jajodia. Planar lattice security structures for multi-level replicated databases. In *Seventh IFIP Working Conference on Database Security*, Huntsville, AL, September 1993. To appear.

[BL76] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report MTR-2997, The Mitre Corporation, Bedford, MA, March 1976.

[Bou85] V. Bouchitté. Chordal bipartite graphs and crowns. *Order*, 2:119–122, 1985.

[Com83] Committee on Multilevel Data Management Security, Air Force Studies Board, National Research Council, Washington, DC. *Multilevel Data Management Security*, 1983.

[Cos92] Oliver Costich. Transaction processing using an untrusted scheduler in a multilevel database with replicated architecture. In Carl Landwehr and Sushil. Jajodia, editors, *Database Security V: Status and Prospects*, pages 173–190. North Holland, 1992.

[Den82] Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley, Reading, MA, 1982.

[DRW82] D. Duffus, I. Rival, and P. Winkler. Minimizing setups for cycle-free ordered sets. *Proceedings of the American Mathematical Society*, 85(4):509–513, August 1982.

[GG78] M.C. Golumbic and C.F. Goss. Perfect elimination and chordal bipartite graphs. *Journal of Graph Theory*, 2:155–163, 1978.

[HC86] M. Hsu and A. Chan. Partitioned two-phase locking. *ACM Transactions on Database Systems*, 11(4):431–446, December 1986.

[JK90] Sushil Jajodia and Boris Kogan. Transaction processing in multilevel-secure databases using replicated architecture. In *Proceedings of the 1990 Symposium on Research in Security and Privacy*, Oakland, CA, May 1990.

[KK92] I.E. Kang and T.F. Keefe. On transaction processing for multilevel secure replicated databases. In *Proceedings European Symposium on Research in Computer Security*, pages 329–347, Toulouse, France, 1992. Springer-Verlag. Lecture Notes in Computer Science, Volume 648.

[Lam73] B.W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[Moh] R. Mohring. Algorithmic aspects of comparability graphs and interval graphs. In [Riv85]. 41–101.

[Riv85] Ivan Rival, editor. *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets*. D. Reidel Publishing Company, Dordrecht, Holland, 1985.

[Tro81] W.T. Trotter Jr. Stacks and splits of partially ordered sets. *Discrete Math.*, 35:229–256, 1981.