A MUTUAL EXCLUSION PROBLEM NEEDING EXPONENTIALLY MANY TOKENS

EDWARD T. ORDMAN

University of Memphis
COMBINATORICS PREPRINT SERIES

Institute of Combinatorics
Department of Mathematical Sciences
University of Memphis
Memphis, TN

A Mutual Exclusion Problem Needing Exponentially Many Tokens

Edward T. Ordman
Department of Mathematical Sciences
University of Memphis, Memphis, TN 38152 U.S.A.

April 16, 1998

Abstract. The *Excluded Taxpayer Problem* is a mutual exclusion problem which shows a limitation of using token-passing methods to administer mutual exclusion. Enforcing the mutual exclusion required by this problem can readily be done in sublinear time and linear space using shared variables, but if token-passing is used, it requires a number of tokens which is exponential in the number of processes. The proof is by making a connection between generalized mutual exclusion problems and coverings of hypergraphs, and using a construction on hypergraphs analogous to a standard result on excluded subgraphs of threshold graphs.

Keywords: Mutual exclusion, token passing, semaphore, threshold graph, hypergraph.

Contact information:
e-mail: ETORDMAN@MEMPHIS.EDU
To May 8, 1998: U. of Memphis, phone +1-901-678-2482, fax 678-2480.
After May 8: 17 Everett Park, New London, NH 03257,
phone +1-603-526-6428 (On sabbatical 1998-99; e-mail is best)

1 Introduction

The *Excluded Taxpayer Problem* is a mutual exclusion problem which shows a limitation of using token-passing methods to administer mutual exclusion. Enforcing the mutual exclusion required by this problem can readily be done in sublinear time and linear space using shared variables, but if token-passing is used, it requires a number of tokens which is exponential in the number of processes. The problem was introduced in [9] but the proof there was much longer and worked only in the case when all tokens required by a given process were distinct.

Definition. Mutual Exclusion. In a collection of computer processes, it may be that not all the processes can be in their *critical section* at one time. For example, several processes want to use the printer but only one at a time can do so. We say that a set S of processes is *mutually excluding* if not all processes in the set may be active at one time. For instance, in the *Dining Philosophers* problem, several philosophers sit around a table; adjacent philosophers share a fork and cannot eat at one time; hence each pair of adjacent philosophers is a minimal mutually excluding set of processes. We suppose that every superset of a mutually excluding set is mutually excluding. By a *mutual exclusion problem*, we mean a set S of processes and a collection $E = \{E_1, E_2, \ldots, E_k\}$ of mutually excluding subsets of S. One standard problem of distributed programming is to provide algorithms that will allow sets of processes to cooperate to ensure that no mutually excluding set of processes are in their critical sections simultaneously.

Notation. Multisets. A *set* has no repeated elements, while a *multiset* may have repeated elements: $\{a, b, c\}$ is a set of three elements and $\{a, b, b, c\}$ is a multiset of four elements, with three distinct elements. We use a * to distinguish multiset operators from set operators, so that $\{a, b\} \cup^* \{b, c\} = \{a, b, b, c\}$ and it is *false* that $\{a, b, b, c\} \subseteq^* \{a, b, c\}$. The repeated elements of a multiset are said to be *of the same type* or *indistinguishable* from one another.

Definition. Token Systems. Given a set of processes $S = \{S_1, S_2, \ldots, S_n\}$ and a mutual exclusion problem on S expressed as a collection $E = \{E_1, E_2, \ldots, E_k\}$ of mutually excluding subsets of S, a *token system* for (S, E) consists of a multiset $T = \{t_1, t_2, \ldots, t_m\}$ of (not necessarily distinct) *tokens* together with a set $\{T_1, T_2, \ldots, T_n\}$ of multisets, one for each S_i, such that each $T_i \subseteq^* T$ and such that a subset $\{S_a, S_b, \ldots, S_c\}$ of S contains no mutually excluding set if and only if $T_a \cup^* T_b \cup^* \ldots \cup^* T_c \subseteq^* T$.

Informally, there is a bowl of tokens t_i, some of which may be indistinguishable from one another; and each process has a list (multiset) of tokens it must obtain from the bowl before in can enter its critical section. If a set of processes can all take the tokens they want from the bowl, they can all proceed at the same time; if there are not enough tokens of some type in the bowl, they cannot.

For a large collection of mutual exclusion problems, see [1]. For another illustration of the way processes may pass tokens to one another, see [7]. For more on the need for multisets, see [9].

A very simple way of administering a bowl of tokens, if they are indistiguishable from one another, is by the use of semaphore operations. In many distributed algorithms, the tokens are never placed "in a bowl" but are passed from one process to another. The bowl abstraction is exactly that, an abstraction to simplify the presentation. To that end, in this paper we ignore the question of how the tokens are "removed from the bowl". We merely consider the question of how many tokens may be needed. Our object is to state a mutual exclusion problem which is not obviously intractable (in fact, which is easy to solve using another means of communication) and for which the smallest token system requires a number of tokens exponential in the size of the problem.

2 The Excluded Taxpayer Problem

The name of this problem is suggested by the Washington, D.C., joke, *Don't tax you, don't tax me, tax the fellow behind the tree.* Suppose we have a legislature containing n members, each of whom represents one of k interest groups. If at least one representative of each interest group is present at a meeting, the debate will be interminable and no tax plan can be passed. However, if all representatives of interest group i are absent, the members present can agree to tax group i. The mutual exclusion problem is to ensure that no legislator enters the room (no process enters its critical section) if her entrance would mean all groups would be represented.

Assuming that k divides n and exactly n/k members represent each group, we denote representative i from group j by $R_{j,i}$. (For concreteness, consider $n = 300$ and $k = 100$, with 3 representatives for each of 100 interest groups.) Process $R_{j,i}$ may enter the meeting (its critical section) if and only if there is some t, $t \neq j$, such that *no* $R_{t,s}$ is present. Clearly, the minimal mutually excluding sets consist of exactly k members, one from each group.

Section 3 gives a shared-variable algorithm enforcing mutual exclusion which is linear in space and sublinear in time. Section 4 shows that any token system to enforce this mutual exclusion will involve exponentially many tokens. In fact, with n members evenly divided into k groups, any minimal token system has $(k-1)(n/k)^k$ tokens.

3 A linear time (shared memory) solution

We first argue that this is not an unreasonably hard mutual exclusion problem. Here is a very simple algorithm. Suppose at the entrance to the legislative chamber we place a chalkboard (shared memory) with k labelled rows, each with a space for n/k marks, and a place for noting the number of groups for whom a representative is already present. Each legislator, before entering the legislative chamber, checks the chalkboard. If a member of her group is in, she adds her mark in the appropriate row and goes in. If no member of her group is in, she checks to be sure that at least one other group is unrepresented and then adds her mark, increments the number of groups represented, and goes in. If hers is the only group unrepresented, she must go away and try later. On exiting the chamber, any legislator must erase one mark from her group's row and decrement the count of groups if she was the only representative of her group present.

Here is a more formal description of the algorithm executed by each process $R(j,i), j = 1\ldots100, i = 1\ldots3$. This algorithm uses integer shared variables occupying at most $n + \log n + 1$ bits, and runs in time $O(\log n)$ (assuming we can look in an array, increment or decrement an integer of size n, or test it for being zero, in time $\log n$).

(blank space to allow algorithm to start on next page)

```
Algorithm: **Excluded Taxpayer, with shared memory**
    Global integer $N$ *initally* 0; { *number of groups represented* }
    Global array $A[1..100]$ of 0..3 initially all 0;
**Begin** process R[j,i]:
    Local boolean $E$          { *TRUE if process may enter* }
{ *entry protocol* }
**repeat**
    Lock($N, A$);          { *guarantees unique access to data* }
    **if** $A[j] > 0$ **then begin** $A[j] := A[j] + 1$; $E :=$ TRUE; **end**
        **else if** $N < 99$ **then**
            **begin** $N := N + 1$; $A[j] := 1$; $E :=$ TRUE; **end**
        **else** $E :=$ FALSE;          {*must wait and try later*};
    Unlock($N, A$);
**until** $E$;
...
{ *exit protocol* }
    Lock($N, A$)
    $A[j] := A[j] - 1$;
    **if** $A[j] = 0$ **then** $N := N - 1$;
    exit;
    Unlock($N, A$)
End.
```

4 The need for exponentially many tokens

The minimal mutually excluding sets in the Excluded Taxpayer problem are precisely the sets of k processes, one from each interest group. There are $(n/k)^k$ such sets. Our goal in this section is to determine the size of a token system necessary to enforce mutual exclusion using tokens. First, we construct a token system that will clearly do the job. For each of the $(n/k)^k$ sets just described, create $k - 1$ tokens, for a total of $(k - 1)(n/k)^k$ tokens. These form a multiset: the $k - 1$ tokens for a given minimal mutually excluding set are indistinguishable from one another, but distinguishable from the tokens for any other minimal mutually excluding set (for example, subscript each token with the name of the set that caused its creation.) Each process R[j,i] will demand a set (not a multiset!) of tokens $T_{j,i}$ in order to enter its critical section: this will include one of the tokens for each minimal mutually excluding set of which R[j,i] is a member.

Observation: The token system just described will enforce mutual exclusion for the Excluded Taxpayer Problem.

The proof is immediate: Any minimal mutually excluding set consists of k members competing for $k-1$ tokens. So at most $k-1$ members of the set can enter at a time, and the mutual exclusion condition will never be violated. Conversely, given any set of processes *not* containing a minimal mutually excluding set, at most $k-1$ want any given type of token, and $k-1$ copies of each type of token are available.

We must now prove that no token system with fewer than $(k-1)(n/k)^k$ tokens can enforce this mutual exclusion. We will use the notation of hypergraphs (see [2]) and of threshold graphs (see [3, 6, 10]), rather loosely, but will give the necessary terminology and lemmas here.

Definition. Hypergraphs. A *hypergraph* consists of a set of points called *vertices* and a collection of non-empty sets of vertices called *hyperedges.* A vertex is called *isolated* if it is in no hyperedge. In this paper we will consider the edge set of a hypergraph to be upward-closed, i.e., any superset of a hyperedge is a hyperedge.

Definition. Threshold Hypergraphs. A hypergraph H is called a *threshold hypergraph* if there is an integer t called the *threshold separator* and an integer label t_v associated with each vertex v, such that a subset of the vertices of H form a hyperedge if and only if the sum of their labels is greater than t. All our threshold hypergraphs will have all labels positive and no isolated vertices.

Definition. Subhypergraphs and Coverings. If H is a hypergraph and J is a hypergraph such that every vertex of J is a vertex of H and every hyperedge of J is a hyperedge of H, then J is called a *subhypergraph* of H. If $H^* = \{H_1, \dots, H_p\}$ is a collection of subhypergraphs of H such that every hyperedge of H contains a hyperedge of some H_i, then H^* is a *hypergraph covering* of H. If also every H_i is a threshold hypergraph, then H^* is a *threshold hypergraph covering* of H.

Now, given a set of processes, a mutual exclusion problem, and a token system, we construct a hypergraph and a threshold hypergraph covering as follows: The vertices of the hypergraph are the processes and the hyperedges are the mutually excluding sets of processes (the minimal hyperedges are the minimal mutually excluding sets of processes). For each type of token v in the token set T, consider the subhypergraph H_v of H which has as vertices those processes wanting one or more copies of token type v to enter their critical section; label each such vertex with the number of copies of v it wants; for the threshold separator t of H_v take the number of copies of v in the set T. We will disregard any token types v that do not ever exclude a process from its critical section, so every vertex of H_v is a member of at least one hyperedge of H_v.

Lemma 4.1 *In the system just described, the H_v form a threshold hypergraph covering of the hypergraph H.*

PROOF: Clearly each H_v is a threshold hypergraph; further, each hyperedge of an H_v consists of processes wanting more than t copies of v and T has only t copies, so they form a mutually excluding set of processes (and a hyperedge of H). To see that any mutually excluding set of processes of H contains a hyperedge of some H_v, suppose that some mutually excluding set of processes $U = \{S_a, \ldots, S_e\}$ does not. This means that for every type of element v of T, the processes in U together need no more copies of v than occur in T. But in that case, they can all obtain the tokens they need, and all of them can enter their critical sections at once. □

Hence, given a mutual exclusion problem and a token system that will enforce that mutual exclusion, there is a threshold hypergraph covering of the underlying mutual exclusion hypergraph. The converse construction works also. Given a covering of a mutual exclusion hypergraph by threshold subhypergraphs, construct a token system as follows: for each threshold subhypergraph V create t tokens of a single type denoted V, and for each vertex of subhypergraph V whose label is s let the corresponding process require s copies of token V to enter its critical section. This establishes a one-to-one correspondence between token types and threshold subhypergraphs, proving the following:

Theorem 4.2 *The minimum number of types of token required to solve a mutual exclusion problem is equal to the minimum number of subhypergraphs in a threshold hypergraph covering of the mutual exclusion hypergraph.*

Similarly, consider a threshold hypergraph covering of a mutual exclusion hypergraph and consider the sum of the threshold separators of the subhypergraphs; this sum equals the number of tokens. Analogous to the above, we have:

Theorem 4.3 *The number of tokens (counting duplicates) needed to solve a mutual exclusion problem is equal to the minimum sum of the threshold separators of the graphs in a threshold hypergraph covering of the mutual exclusion hypergraph (taken over all such threshold hypergraph coverings).*

In general, finding minimal threshold subhypergraph coverings is highly intractable (it is even NP-complete to determine if a graph can be covered by two threshold graphs [4]). However, for the Excluded Taxpayer Problem, the structure of the threshold hypergraphs which are subhypergraphs of the given hypergraph are particularly simple.

Hereafter we let $\mathcal{E}$ denote the mutual exclusion hypergraph of the Excluded Taxpayer Problem. The minimal hyperedges of this hypergraph are precisely the k-vertex sets with one element from each interest group; thus all minimal hyperedges have exactly k vertices and there are $(n/k)^k$ such hyperedges. We now set out to characterize the subhypergraphs of $\mathcal{E}$ which are threshold hypergraphs.

Lemma 4.4 *Let F be a subhypergraph of the Excluded Taxpayer hypergraph $\mathcal{E}$ and suppose F is a threshold hypergraph. Then there is at most one interest group j such that F contains more than one of the processes $R_{j,i}$ for $i = 1, \ldots, n/k$.*

PROOF: Suppose (by relabelling as needed) that interest groups 1 and 2 each have at least two members which are vertices of hyperedges in F: choose the two from each with the largest labels and call them $R_{1,1}$, $R_{1,2}$, $R_{2,1}$, and

$R_{2,2}$. Suppose (in the threshold labelling) that $R_{j,i}$ has label $r_{j,i}$ and suppose that $r_{1,1} \geq r_{1,2}$ and $r_{1,1} \geq r_{2,1} \geq r_{2,2}$. Since F has no isolated vertices and all its minimal hyperedges contain representatives of all interest groups, there are vertices forming minimal hyperedges containing each of these four vertices; in particular there is a hyperedge of the form $\{R_{1,2}, R_{2,c}, R_{3,f}, \ldots, R_{k,g}\}$ for some $c, f, \ldots, g$ in $1, \ldots, n/k$. This set of vertices has label sum exceeding t. Therefore, so does the set $R^* = \{R_{1,2}, R_{1,1}, R_{3,f}, \ldots, R_{k,g}\}$ since $r_{1,1} \geq r_{2,1} \geq r_{2,c}$. However, R^* is not a hyperedge of $\mathcal{E}$, so it cannot be a hyperedge of F; it contains no representative of group 2. This contradiction shows that if F is a threshold hypergraph and a subgraph of $\mathcal{E}$, it cannot meet two interest groups in two or more vertices each. □

Thus we see that a threshold subhypergraph of $\mathcal{E}$ includes at most all n/k representatives of one interest group and exactly one representative of each other interest group. Thus it includes at most n/k distinct minimal hyperedges of $\mathcal{E}$. Since $\mathcal{E}$ has $(n/k)^k$ minimal hyperedges, this proves:

Theorem 4.5 *Any token system for the Excluded Taxpayer Problem (with n representatives evenly divided into k groups) requires at least $(n/k)^{k-1}$ distinct types of tokens.*

To see that the above number can be attained, and determine the sum of the threshold separators, we must examine the structure (and labelling) of the threshold subhypergraphs of $\mathcal{E}$ in more detail.

Lemma 4.6 *Let F be a threshold subhypergraph of $\mathcal{E}$ and suppose F contains p vertices from interest group number q (and one vertex from each of the other $k-1$ interest groups). The smallest labels that will produce this are: 1 for each vertex of F in group q and p for each vertex of F outside group q, with a threshold separator of $t = p(k-1)$.*

PROOF: Since all vertex labels in a threshold hypergraph are at least 1, the vertices in group q must have label at least 1. Let $m \neq q$ and suppose that the following processes of F are in their critical sections: all those from group q and one from each other group except m (This is possible as this set contains no process from group m). Now suppose $p-1$ processes from group q exit their critical sections, returning their tokens (at least $p-1$) to the pool. It is still not possible for the process in group m to enter its critical section (one process from group q is still there) so that process must require more than $p-1$ tokens to enter. Hence each process of F outside group q requires at least p tokens. But since all $k-1$ processes outside group q can enter their critical sections at once, there must be at least $p(k-1)$ tokens available. Finally, with this labelling, it is easy to check that any $k-1$ vertices outside group q, or up to p vertices from group q and $k-2$ vertices from outside group q, have label sums not over $p(k-1)$, but any larger subset of F has label sum exceeding $p(k-1)$ (and represents a set of processes containing a miminal mutually excluding set). □

This result has a surprising corollary: a threshold subhypergraph F of $\mathcal{E}$ has a threshold separator of at least $p(k-1)$ provided it contains at least p minimal hyperedges of $\mathcal{E}$. But that is exactly what we needed to show that the originally constructed token system for the Excluded Taxpayer Problem is minimal in the total number (although not in the number of types) of tokens:

Theorem 4.7 *Any token system for the Excluded Taxpayer Problem (with n representatives evenly divided into k groups) requires at least $(k-1)(n/k)^k$ tokens.*

5 Open questions

We have shown that token-passing is in some cases substantially less efficient than using shared-variable algorithms for enforcing mutual exclusion.

There are a number of algorithms in the literature for managing token-passing, many of which also consider fairness, deadlock, and other considerations beyond mutual exclusion. Most of these assume that the processes are competing for a single token (simple mutual exclusion), for single copies of a single type of token (the k-of-n entry problem, e.g. [7, 12]), or form a system modeled by a graph rather than a hypergraph [5]. In [11] a method is given to formalize the "bowl of tokens" entry algorithm discussed in this paper to guarantee freedom from deadlock and bounded waiting time, for an arbitrary mutual exclusion problem, using only semaphores (that is, **P** and **V** operations) as the programming tools. Since this method may, however, lead to exponentially long waiting times, there is much room for finding better mutual exclusion algorithms for even the case of a threshold hypergraph (one token type with different processes wanting different numbers of tokens), as well as for mutual exclusion problems modelled by more general hypergraphs.

Very little is known about threshold hypergraphs. While the theory of threshold graphs is fairly well developed, the lack even of simple characterizations or recognition algorithms for threshold hypergraphs makes using them difficult. For example, in [8] the author showed that for mutual exclusion graphs, the smallest token system had a number of tokens equal to the clique covering number of the graph. This depended on the fact that the threshold separator of a graph is at least equal to its clique covering number. The most direct extension to threshold hypergraphs fails; can anything be salvaged? Can a threshold hypergraph be recognized in NP time? Are there interesting classes of hypergraphs for which minimal token systems can be found?

References

[1] G. Andrews, "Concurrent Programming: Principles and Practice", Benjamin/Cummings, Redwood City, Ca., 1991.

[2] C. Berge, "Graphs and Hypergraphs" , North-Holland Publishing Company, New York, 1973.

[3] V. Chvátal and P. Hammer, *Aggregation of inequalities in integer programming*, Ann. Discrete Math. 1 (1977), 145–162.

[4] M. B. Cozzens and R. Liebowitz, *Threshold dimension of graphs*, SIAM J. Algebraic Discrete Methods 5(1984), 579–595.

[5] E. W. Dijkstra, *Solution of a problem in concurrent programming control*, Comm. ACM 8(1965), 569.

[6] P. Henderson and Y. Zalcstein, *A graph theoretic characterization of the* $\mathbf{PV}_{\mathbf{chunk}}$ *class of synchronizing primitives*, SIAM J. Comp. 6 (1977), 88–108.

[7] M. Naimi, *Distributed algorithm for K-entries to a critical section based on the directed graphs*, Operating Systems Review, 27(October 1993).

[8] E. T. Ordman, *Threshold coverings and resource allocation*, Proc. 16th Southeastern International Conf. on Graph Theory, Combinatorics, and Computing, Congr. Numer. 49(1985), 99–113.

[9] E. T. Ordman, *Cliques in hypergraphs and mutual exclusion using tokens*, J. Combinatorial Mathematics and Combinatorial Computing 19(1995), 29-224.

[10] E. T. Ordman, *Minimal threshold separators and memory requirements for synchronization*, SIAM J. Computing 18(1989), 152-165.

[11] E. T. Ordman, E. Eberbach, A. Anwar, *Mutual exclusion with semaphores only*, in preparation.

[12] P. K. Srimani and R. L. N. Reddy, *Another distributed algorithm for multiple entries to a critical section*, Inform. Process. Lett. 41(1992), 51–57.