IEICE TRANS. INF. & SYST., VOL.E82–D, NO.5 MAY 1999

PAPER

Coterie for Generalized Mutual Exclusion Problem*

Shao Chin SUNG†, ***Nonmember and*** **Yoshifumi MANABE**††, ***Member***

SUMMARY This paper discusses the *generalized mutual exclusion problem* defined by H. Kakugawa and M. Yamashita. A set of processes shares a set of resources of an identical type. Each resource must be accessed by at most one process at any time. Each process may have different accessible resources. If two processes have no common accessible resource, it is reasonable to ensure a condition in resource allocation, which is called *allocation independence* in this paper, i.e., resource allocation to those processes must be performed without any interference. In this paper, we define a new structure, *sharing structure coterie*. By using a sharing structure coterie, the resource allocation algorithm proposed by H. Kakugawa and M. Yamashita ensures the above condition. We show a necessary and sufficient condition of the existence of a sharing structure coterie. The decision of the existence of a sharing structure coterie for an arbitrary distributed system is NP-complete. Furthermore, we show a resource allocation algorithm which guarantees the above requirement for distributed systems whose sharing structure coteries do not exist or are difficult to obtain.

key words: *distributed systems, coterie, resource allocation algorithm*

1. Introduction

In many *distributed systems*, processes share resources such as files, memory, and printers. When *mutual exclusion* in access to resources must be ensured, each resource must not be accessed by more than one process at the same time. The problem to allocate resources in such a distributed system is called *mutual exclusion problem*.

Many distributed algorithms have been presented for *k-mutual exclusion problem*, in which there are k (≥ 1) shared resources of identical type and all shared resources are accessible to every process. Garcia-Molina and Barbara [2] introduced *coteries*, and showed a coterie-based algorithm for the 1-mutual exclusion problem which corresponds to the case of one shared resource. Fujita et al. [1] and Manabe et al. [6] considered the k-mutual exclusion problem for arbitrary fixed k. They defined different generalizations of coteries and showed algorithms for the k-mutual exclusion problem based on each one.

In this paper, we consider a problem called the *generalized mutual exclusion problem*, in which each process may have different accessible resources. This problem is defined by Kakugawa and Yamashita [4]. They proposed an algorithm for this problem based on a new class of coteries, called *local coteries*.

In the generalized mutual exclusion problem, if two processes have no common accessible resource, it is reasonable to ensure a condition in resource allocation, which is called *allocation independence* in this paper, i.e., resource allocation to those processes must be performed without any interference. However, the algorithm proposed in [4] does not ensure this condition.

In order to ensure the allocation independence, we introduce a new class of coteries, called *sharing structure coteries*. We show that by using a sharing structure coterie instead of a local coterie, the algorithm proposed by Kakugawa and Yamashita [4] ensures the allocation independence. Then, we consider existence of a sharing structure coterie for any given distributed system. We show a necessary and sufficient condition of the existence, where decision problem of the existence is NP-complete.

Furthermore, we show an algorithm for the generalized mutual exclusion problem which ensures the allocation independence, for distributed systems whose sharing structure coteries do not exist or are difficult to obtain.

2. Preliminaries

The model of *distributed systems* and the *generalized mutual exclusion problem* are defined as follows.

A *distributed system* $\mathcal{S}$ is a 3-tuple $(\mathcal{U}, \mathcal{R}, \alpha)$ **: $\mathcal{U}$ is a set of processes. Each process has its own local clock. Every two processes are connected by a bidirectional communication link. Information exchange between the processes is based on message-passing through the links. The delivery of messages may have unpredictable finite delay, but the order of messages is unchanged (i.e., FIFO). All processes and

Manuscript received August 12, 1998.

†The author is with School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923–1292 Japan.

††The author is with NTT Basic Research Laboratories, Atsugi-shi, 243–0198 Japan.

*Part of this work was done while the first author was staying at NTT Basic Research Laboratories. The preliminary version of this paper was presented at IPSJ International Symposium on Information Systems and Technologies for Network Society (1997).

**In [4], tuple $(\mathcal{U}, \mathcal{R}, \alpha)$ is called the *sharing structure* of a distributed system.

links are assumed to be error-free. $\mathcal{R}$ is a set of resources. Each resource is shared by one or more processes. A resource r is called *accessible* to process u if process u is one of the processes that share resource r. $\alpha : \mathcal{U} \to 2^{\mathcal{R}}$ is a mapping such that $\alpha(u)$ denote the set of all accessible resources of process u. We assume without loss of generality that for every process $u \in \mathcal{U}$ there exists some process $v \neq u$ such that $\alpha(u) \cap \alpha(v) \neq \emptyset$. Otherwise, resource allocation for such a process u can be performed independently.

A *generalized mutual exclusion problem* of a distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$ is to allocate resources in $\mathcal{R}$ according to requests from processes in $\mathcal{U}$ such that the following conditions are satisfied.

- *Mutual exclusion*: Each resource is accessed by at most one process at the same time.
- *Allocation validity*: At any time, each process u accesses resource $r \in \mathcal{R}$ only if $r \in \alpha(u)$.

The k-mutual exclusion problem can be defined as the case that $|\mathcal{R}| = k$ and $\alpha(u) = \mathcal{R}$ for all $u \in \mathcal{U}$.

Coterie is introduced in [2] for resource allocation algorithm of 1-mutual exclusion problem. Coterie Q is a family of subset of $\mathcal{U}$, i.e., $Q \subseteq 2^{\mathcal{U}}$, that satisfies the following properties:

- *Non-emptiness*: $\forall q \in Q\ [q \neq \emptyset]$.
- *Minimality*: $\forall q, r \in Q\ [q \not\subseteq r]$.
- *Intersection property*: $\forall q, r \in Q\ [q \cap r \neq \emptyset]$.

An element of a coterie is called a *quorum*. The coterie-based algorithm for the resource allocation problem [5], [9] can simply described as follows: Determine a coterie Q. Initially, each process has one "permission." If a process u wants to access the resource, it arbitrarily selects a quorum $q \in Q$, and sends a request to each process in q. Then, u waits to receive a permission from each process in q, and it accesses the resource. After the access, u releases the resource and returns the permission to each process in q. The intersection property ensure the mutual exclusion condition.

To ensure that the algorithm is deadlock-free and starvation-free, a priority of requests is introduced [5], i.e., the request with the smallest timestamp has the highest priority. Suppose a process u had sent its permission to a process v's request whose timestamp is T_v, and v has not received enough permissions, i.e., v has not accessed the resource. If u receives a request from a process w which has timestamp $T_w < T_v$, then u will ask v to return the permission, and u sends the permission to w.

Subsequently, coteries are extended for k-mutual exclusion problem for arbitrary fixed k by Fujita et al. [1] and Manabe et al. [6].

In the generalized mutual exclusion problem, each process may have a different set of accessible resources, i.e., it is allowed that $\alpha(u) \neq \alpha(v)$ for any $u, v \in \mathcal{U}$. Kakugawa and Yamashita [4] introduced a new class of coteries, called *local coteries*. A local coterie $\{Q_u \subseteq 2^{\mathcal{U}} \mid u \in \mathcal{U}\}$ for a distributed system $\mathcal{S}$ satisfies the following properties:

- *Non-emptiness*: $\forall u \in \mathcal{U}\ [Q_u \neq \emptyset]$.
- *Minimality*: $\forall u \in \mathcal{U}, \forall q, r \in Q_u\ [q \not\subseteq r]$.
- *Intersection property*: $\forall u, v \in \mathcal{U}$,

$$[\alpha(u) \cap \alpha(v) \neq \emptyset \ \Rightarrow\ \forall q \in Q_u, \forall r \in Q_v\ [q \cap r \neq \emptyset]].$$

They showed an algorithm for constructing a local coterie for an arbitrary distributed system $\mathcal{S}$.

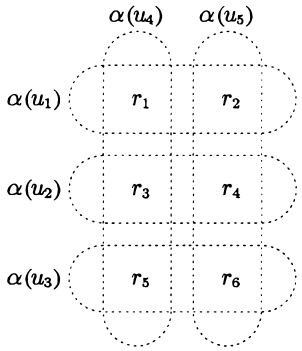


Fig. 1 Distributed system $\mathcal{S}$ in Example 1.

Example 1: Consider a distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$, where $\mathcal{U} = \{u_1, u_2, \ldots, u_5\}$, $\mathcal{R} = \{r_1, r_2, \ldots, r_6\}$, and

$\alpha(u_i) = \{r_{2i-1}, r_{2i}\}$ for $i = 1, 2, 3$,
$\alpha(u_4) = \{r_1, r_3, r_5\}$, and
$\alpha(u_5) = \{r_2, r_4, r_6\}$ (see Fig. 1).

A local coterie $\{Q_u \subseteq 2^{\mathcal{U}} \mid u \in \mathcal{U}\}$ for $\mathcal{S}$ obtained by the algorithm in [4] is shown in the following.

$Q_{u_i} = \{\{u_i, u_4, u_5\}\}$ for $i = 1, 2, 3$, and
$Q_{u_i} = \{\{u_i, u_1, u_2, u_3\}\}$ for $i = 4, 5$.

The coterie-based algorithm which uses a local coterie for the generalized mutual exclusion problem in [4] is similar to the one for the mutual exclusion problem. Instead of coterie, determine a local coterie $\mathcal{Q} = \{Q_u \subseteq 2^{\mathcal{U}} \mid u \in \mathcal{U}\}$. Instead of a permission, a state list is sent to the requesting process, where the state list shows that each resource is free or not at that time. Process u has the right to access to a set of resources $\mathcal{R}_u \subseteq \alpha(u)$ if the following conditions are satisfied:

- Process u receives a state list from each process in an arbitrary quorum $q \in Q_u$.
- States of all resources in $\mathcal{R}_u$ are free in every state list.

When these conditions are satisfied and u wants to access the resources, process u updates the state lists and sends it back to each process in q. When process u finished the access of resources, process u sends a message to each process in q to tell them the access is finished.

However, unnecessary waiting for resource alloca-

tion among processes might occur. In Example 1, let us consider the case that only u_1 and u_2 want to access the resources at the same time. Since $\alpha(u_1) \cap \alpha(u_2) = \emptyset$, these two requests do not block each other. However, $q_{u_1} \cap q_{u_2} = \{u_4, u_5\} \neq \emptyset$. Thus, u_4 and u_5 first send the state list to the higher priority request, say, u_1. After the state list update by u_1, the updated state list is sent to u_2 and u_2 can access the resources. When there exist k resources, there can be such a process waiting chain whose length is $O(k)$, and this waiting is unnecessary. It is natural to ensure that there is no such unnecessary waiting, i.e., the resource allocation must also satisfy the following condition.

- *Allocation independence:* If $\alpha(u) \cap \alpha(u') = \emptyset$ for $u, u' \in \mathcal{U}$, then resource allocation to u and u' must be performed without any interference.

3. Sharing Structure Coteries

In this section, we define a new class of coteries, called *sharing structure coteries*, which is a subclass of local coteries. By using a sharing structure coterie instead of a local coterie, the algorithm proposed in [4] becomes to ensure the allocation independence condition.

A *sharing structure coterie* satisfies the three properties of a local coterie (i.e., non-emptiness, minimality, and intersection property) and the following additional property:

- *Disjointness property*: $\forall u, v \in \mathcal{U}$,

$$[\alpha(u) \cap \alpha(v) = \emptyset \Rightarrow \forall q \in Q_u, \forall r \in Q_v \ [q \cap r = \emptyset]].$$

It is clear that if a local coterie satisfies this property, then the unnecessary waiting mentioned in the previous section does not occur.

Unfortunately, there exist some distributed systems that have no sharing structure coteries. In the following, we consider the necessary and sufficient condition of the existence of a sharing structure coterie for an arbitrary distributed system.

We show that the following decision problem is NP-complete. Given an arbitrary distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$,

Is there a sharing structure coterie for $\mathcal{S}$?

In the following, we show that this decision problem is equivalent to an NP-complete problem called "COVERING BY CLIQUE" [3].

Definition 1: A *clique cover* of a graph $G = (V, E)$ is a collection of subsets $V_1, \ldots, V_k$ of V such that,

- each V_i induces a complete subgraph of G, and
- for each edge $\{u, v\} \in E$, there exists some V_i that contains both u and v.

Given a graph G and a positive integer $K \leq |E|$, the "*COVERING BY CLIQUE*" problem is:

Is there a clique cover of G with cardinality $k \leq K$?

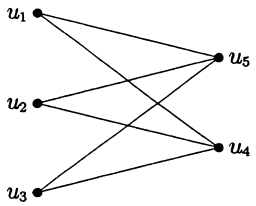


Fig. 2 Sharing graph $G_{\mathcal{S}}$ of $\mathcal{S}$.

Definition 2: For an arbitrary distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$, sharing graph $G_{\mathcal{S}} = (\mathcal{U}, E_{\mathcal{S}})$ of $\mathcal{S}$ is an undirected graph such that

$$E_{\mathcal{S}} = \{\{u, v\} \in \mathcal{U} \times \mathcal{U} \mid u \neq v, \ \alpha(u) \cap \alpha(v) \neq \emptyset\}.$$

A sharing graph of distributed system in Example 1 is shown in Fig. 2. The existence problem of sharing structure coteries is related to "COVERING BY CLIQUE" problem by the sharing graph $G_{\mathcal{S}}$.

Theorem 3: There exists a sharing structure coterie for distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$ if and only if there exists a clique cover of $G_{\mathcal{S}}$ with cardinality at most $|\mathcal{U}|$.

Let $\mathcal{Q} = \{Q_u \subseteq 2^{\mathcal{U}} \mid u \in \mathcal{U}\}$ be a sharing structure coterie for $\mathcal{S}$. Then, $\mathcal{Q}' = \{Q'_u \subseteq Q_u \mid u \in \mathcal{U}, \ |Q'_u| = 1\}$ is also a sharing structure coterie for $\mathcal{S}$. Note that all quorums in $\mathcal{Q}'$ are singletons. We call such a coterie $\mathcal{Q}'$ a *singleton sharing structure coterie*. It is clear that there exists a sharing structure coterie for $\mathcal{S}$ if and only if there exists a singleton sharing structure coterie for $\mathcal{S}$. From the theorem by Ordman [7] † (also see [8]), there exists a singleton sharing structure coterie for $\mathcal{S}$ if and only if there exists a clique cover of $G_{\mathcal{S}}$ which consists of at most $|\mathcal{U}|$ cliques.

For the distributed system $\mathcal{S}$ in Example 1, there is no sharing structure coterie for $\mathcal{S}$, since it can be easily verified that all clique covers of $G_{\mathcal{S}}$ has cardinality at least 6.

Example 2: Consider a distributed system $\mathcal{S}' = (\mathcal{U}', \mathcal{R}', \alpha')$, where $\mathcal{U}' = \{u_1, u_2, \ldots, u_5\}$, $\mathcal{R}' = \{r_1, r_2, \ldots, r_6, r_7\}$, and

$\alpha'(u_i) = \{r_{2i-1}, r_{2i}\}$ for $i = 1, 2, 3$,
$\alpha'(u_4) = \{r_1, r_3, r_5, r_7\}$, and
$\alpha'(u_5) = \{r_2, r_4, r_6, r_7\}$.

The sharing graph $G_{\mathcal{S}'}$ for $\mathcal{S}'$ is shown in Fig. 3. Then, $\{\{u_1, u_4, u_5\}, \{u_2, u_4, u_5\}, \{u_3, u_4, u_5\}\}$ is a clique cover of $G_{\mathcal{S}'}$. From Theorem 3, a sharing structure coterie $\mathcal{Q} = \{Q_u \mid u \in \mathcal{U}'\}$ for $\mathcal{S}'$ (with the smallest size) can

†The theorem by Ordman [7] is showed for a token system which is equivalent to a distributed system using a singleton sharing structure coterie.

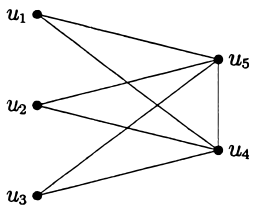


Fig. 3 Sharing graph $G_{\mathcal{S}'}$ of $\mathcal{S}'$ in Example 2.

be defined as follows. $Q_{u_i} = \{q_{u_i}\}$ for $i = 1, 2, \ldots, 5$ such that

- $q_{u_i} = \{u_i\}$ for $i = 1, 2, 3$, and
- $q_{u_i} = \{u_1, u_2, u_3\}$ for $i = 4, 5$.

There also exists some sharing structure coterie $\mathcal{Q} = \{Q_u \mid u \in \mathcal{U}'\}$ for $\mathcal{S}'$ such that some Q_u is not a singleton. One of which can be defined as follows.

- $Q_{u_1} = \{\{u_1, u_4\}, \{u_1, u_5\}, \{u_4, u_5\}\}$, and
- $Q_{u_i} = \{\{u_i\}\}$ for $i = 2, 3$, and
- $Q_{u_i} = \{q \cup \{u_2, u_3\} \mid q \in Q_{u_1}\}$ for $i = 4, 5$.

4. Algorithm for No Sharing Structure Coterie Cases

As shown in Theorem 3, there exists some distributed system which has no sharing structure coterie, and even if there exists such a coterie, it is sometimes very difficult to obtain.

In this section, we consider a resource allocation algorithm for distributed systems whose sharing structure coterie do not exist or is difficult to obtain.

For any distributed system $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \alpha)$, suppose a clique cover $\mathcal{C}$ for the sharing structure graph $G_{\mathcal{S}}$ such that $|\mathcal{C}| > |\mathcal{U}|$ is given. Note that it is not implying that no sharing structure coterie for $\mathcal{S}$ exists. The size of $\mathcal{C}$ need not be minimum for given $\mathcal{S}$. Obtaining such a clique cover is not difficult, since $E_{\mathcal{S}}$ is also a clique cover in which each clique is a set with two elements (i.e., endpoints of an edge).

Consider a distributed system $\mathcal{S}' = (\mathcal{U} \cup \mathcal{V}, \mathcal{R}, \alpha')$, where $\mathcal{U} \cap \mathcal{V} = \emptyset$ and $|\mathcal{U} \cup \mathcal{V}| = |\mathcal{C}|$,

$$\alpha'(u) = \begin{cases} \alpha(u) & \text{if } u \in \mathcal{U}, \\ \emptyset & \text{otherwise.} \end{cases}$$

Then, $\mathcal{C}$ is also a clique cover for sharing graph $G_{\mathcal{S}'}$ of $\mathcal{S}'$, since each $v \in \mathcal{V}$ is an isolated node in $G_{\mathcal{S}'}$. The nodes of $\mathcal{V}$ are imaginary nodes and do not access the resources at all. Thus, Q_v can be $\{\emptyset\}$ for any $v \in \mathcal{V}$.

From Theorem 3, there exists a sharing structure coterie for $\mathcal{S}'$. Thus, by simulating $\mathcal{S}'$ by $\mathcal{S}$, i.e., each process in $\mathcal{V}$ is simulated by a process in $\mathcal{U}$, resource allocation for $\mathcal{S}$ can be performed by the algorithm in [4] using a sharing structure coterie for $\mathcal{S}'$.

5. Message Complexity

The analysis of the message complexity for this algorithm is the same as the one in [4]. In the best case, the message complexity for one request is $4|q|$, where q is the smallest quorum. In the worst case, the message complexity for one request is $(7+|\alpha(u)|)|q'|$, where q' is the largest quorum. In the following, we show that by estimating the size of quorums the number of messages sent using a sharing structure coterie is not larger than that using a local coterie.

First, we estimate the size of quorums of a local coterie. By the construction algorithm for local coteries in [4], the size of process u's quorum, $|q_u|$, satisfies $|q_u| = |\{v \in \mathcal{U} \mid \alpha(u) \cap \alpha(v) \neq \phi\}|$. Precisely, since $u \in q_u$ and u does not have to send a message to itself, the message complexity for one request is $4(|q| - 1)$ in the best case, and is $(7 + |\alpha(u)|)(|q'| - 1)$ in the worst case.

Then, we estimate the size of quorums of a sharing structure coterie. As shown in the previous section, any sharing structure coterie can be constructed from a given clique cover in $G_{\mathcal{S}}$, and the size of u's quorum q'_u, $|q'_u|$, is the number of cliques which contain u in the clique cover. Thus, $|q'_u|$ is at most the degree of u in G_u, i.e., $|q'_u| \leq |\{v \in \mathcal{U} \mid v \neq u, \alpha(u) \cap \alpha(v) \neq \phi\}| < |q_u|$, i.e., the size of quorum in a sharing structure coterie is not larger than that in a local coterie. Therefore, the number of messages sent using a sharing structure coterie is not larger than that using a local coterie.

6. Conclusion

We have defined a new class of coteries, sharing structure coteries, for the generalized mutual exclusion problem. By using a sharing structure coterie, we can construct a resource allocation algorithm which ensures the allocation independence, i.e., resource allocation for two processes with no common accessible resource are performed without any interference. We showed the existence condition of a sharing structure coterie for an arbitrary distributed system.

Furthermore, for distributed systems whose sharing structure coteries do not exist or are difficult to obtain, we showed that the resource allocation can be performed by the algorithm in [4] using a sharing structure coterie for a distributed system with imaginary processes.

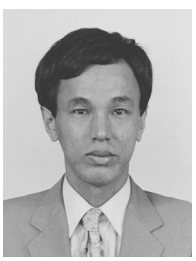
Acknowledgments

The authors would like to thank Prof. Masafumi Yamashita of Kyushu University for his helpful comments. They also thank Dr. Hirofumi Katsuno of NTT Basic Research Laboratories for his encouragement and suggestions.

References

[1] S. Fujita, M. Yamashita, and T. Ae, "Distributed k-mutual exclusion problem and k-coteries," Proc. 2nd International Symposium on Algorithms, LNCS 557, pp.22–31, 1991.
[2] H. Garcia-Molina and D. Barbara, "How to assign votes in a distributed system," J. ACM, vol.32, no.4, pp.841–860, 1985.
[3] M.R. Garey and D.S. Johnson, "Computers and Intractability: A guide of the theory of NP-completeness," W.H. Freeman and Company, San Francisco, 1979.
[4] H. Kakugawa and M. Yamashita, "Local coteries and a distributed resource allocation algorithm," Trans. Information Processing Society of Japan, vol.37, no.8, pp.1487–1498, 1996.
[5] M. Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," ACM Trans. Computer Syst., vol.3, no.2, pp.145–159, 1985.
[6] Y. Manabe, R. Baldoni, M. Raynal, and S. Aoyagi, "k-Arbiter: A safe and general scheme for h-out of-k mutual exclusion," Theoretical Computer Science, vol.193, no.1-2, pp.97–112, Feb. 1998.
[7] E.T. Ordman, "Threshold coverings and resource allocation," Congressus Numerantium, vol.49 , pp.99–113, 1985.
[8] E.T. Ordman, "Limitations of using tokens for mutual exclusion," Proc. Computer Science, pp.26–29, Feb. 1995.
[9] M. Singhal and N.G. Shivaratri, "Advanced Concepts in Operating Systems– Distributed, Database, and Multiprocessor Operating Systems," McGraw-Hill, 1994.

Shao Chin Sung received B.S. degree from Waseda University in 1993, and M.S. and Ph.D. degrees from the Japan Advanced Institute of Science and Technology (JAIST) in 1995 and 1998, respectively. He is currently a Research Associate at JAIST. His research interests include distributed algorithms, circuit complexity theory, and quantum algorithms.



Yoshifumi Manabe received B.E., M.E., and Ph.D. degrees in information engineering from Osaka University, Japan in 1983, 1985, and 1993, respectively. In 1985, he joined Nippon Telegraph and Telephone Corporation (NTT). In 1994-95, he was a visiting researcher in Department of Computer Science, The Johns Hopkins University. He is currently a senior research scientist in NTT Basic Research Laboratories. His research interests are distributed algorithms, graph theory, and real-time systems. He is a member of the ACM, the Japan Society for Industrial and Applied Mathematics, and the Information Processing Society of Japan.