DOCUMENT RESUME

ED 090 982 IR 000 593

AUTHOR	Hsiao, D. K.; And Others
TITLE	Context Protection and Consistent Control in Data Base Systems (Part I).
INSTITUTION	Ohio State Univ., Columbus. Computer and Information Science Research Center.
SPONS AGENCY	Office of Naval Research, Washington, D.C.
REPORT NO	OSU-CISRC-TR-/3-9
PUB DATE	Feb 74
NOTE	33p.
EDRS PRICE	MF-$0.75 HC-$1.85 PLUS POSTAGE
DESCRIPTORS	*Computer Programs; Computer Science; *Confidentiality; *Data Bases; Data Processing; Information Storage; Input Output Devices; Time Sharing
IDENTIFIERS	*Data Protection Systems; Data Security Systems

ABSTRACT

Although the indispensibility of access control mechanisms in data secure systems has been recognized, there is a need for more subtle protection and refined control mechanisms. This report describes context protection mechanisms which enable the same data unit to be protected differently in different contexts, the differences being determined by the manner in which the fields and records are being accessed. The study shows that the method can be enforced by means of certain built-in relations among the data units involved. Three basic and primary relations are proposed as a starting point for more elaborate control systems. These primary relations allow the necessary and sufficient conditions under which violations will occur to be identified. With these conditions and the security system, it becomes possible to propose a data definition language for specifying data base protection requirements and to develop an access control mechanism for enforcing the requirements. (WDR)

ED 090982

TECHNICAL REPORT SERIES

COMPUTER & INFORMATION SCIENCE RESEARCH CENTER

THE OHIO STATE UNIVERSITY COLUMBUS, OHIO

ED 090982

OSU-CISRC-TR-73-9

CONTEXT PROTECTION AND CONSISTENT CONTROL IN DATA BASE SYSTEMS (PART I)

by

D. K. Hsiao, D. S. Kerr and C. J. Nee

Work performed under

Contract N00014-67-A-0232-0022, Office of Naval Research

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRO-
DUCED EXACTLY AS RECEIVED FROM
THE PERSON OR ORGANIZATION ORIGIN-
ATING IT. POINTS OF VIEW OR OPINIONS
STATED DO NOT NECESSARILY REPRE-
SENT OFFICIAL NATIONAL INSTITUTE OF
EDUCATION POSITION OR POLICY

The Computer and Information Science Research Center
The Ohio State University
Columbus, Ohio 43210
February 1974

Preface

The project on Data Security and Data Secure Systems was formed in the Fall of 1972 and funded on March 1, 1973. The principal investigator of the project is Dr. David K. Hsiao, Associate Professor of Computer and Information Science. There are five Graduate Associates and Assistants on the project. They are R. I. Baum, N. Kaffin, E. J. McCauley, C. J. Nee, and S. Peden. Presently, Dr. Douglas S. Kerr, Associate Professor of Computer and Information Science, is serving as an investigator on the project.

Five major research and experimentation efforts are underway. We intend to issue a series of technical reports at the milestones of these efforts. Although early reports may be preliminary, we believe that they can serve as the position papers of the pursuing research. These five research and experimentation efforts are listed as follows:

(1) A data secure system based on the theory of security deadlock.

(2) Theoretical foundations for context protection and consistent control in data secure systems.

(3) A data secure computer (hardware) architecture.

(4) Design and certification of data secure system kernels.

(5) A system for experimenting access control mechanisms.

The report is published by the Computer and Information Science Research Center of The Ohio State University which consists of the staff, graduate students, and faculty of many University departments and laboratories. The research was administered and monitored by The Ohio State University Research Foundation.

Abstract

Although the capability of the access control mechanisms to regulate field, record, and file security has been recognized as indispensible in advanced data secure systems, there is the need of more subtle protection and refined control which we shall call context protection and consistent control. Context protection enables the same data unit (field, record or file) to be protected differently in different contexts. For example, the same data field may be protected differently in different records. The difference may be determined by the manner in which the fields and records are being accessed. Consistent control is concerned with the problem that when new data units based on the old data units of the data base are created by the user, these data units must be protected consistently in the sense that their access attributes must be generated automatically and must conform with the access attributes of the old data units. Our study has begun to show that both context protection and consistent control can be enforced by means of certain built-in relations among the data units involved. These relations under certain conditions can reveal any violation of context protection and consistent control. Here our first step is to identify those relations which are basic and primary to the contextual relations. It is hoped that by proposing these basic and primary relations, more elaborate contextual relations among data elements can be defined for protection reasons. Furthermore, a method of enforcing the protection can be facilitated by these basic and primary relations. Necessary and sufficient conditions under which a protection violation will occur due to contextual security constraints must be identified. With these conditions and the method, it will be possible to propose a data definition language for specifying data base protection requirements and to develop an access control mechanism for enforcing the requirements.

This report presents our first finding in context protection. No discussion on consistent control is included here. A graph-theoretic approach is used. The use of directed graphs is not surprising since graphs are good candidates for representing simple relations.

Table of Contents

1. A Simple Example--the Limitation of Access Matrix

Conceptually, an access control mechanism in a computer system consists of three parts. The first part is a set of objects, an object being an entity to which access must be controlled. The second part is a set of subjects, a subject being an active entity whose access to objects must be controlled. The last part of the access control mechanism consists of the rules which govern the accessing of objects by subjects. Examples of objects are files, records, fields, and programs. User tasks executing in the computer system are examples of subjects.

In this discussion, all the time-variant information specifying the types of access to objects that subjects have is regarded as constituting the access control information of the mechanism. At a given time, the access control information may be represented by an access matrix A, with subjects identifying the rows and objects the columns. The entry A(S,O) contains access attributes, specifying the access rights held by subject S to object O. Below is an example of such an access matrix A, as depicted in [4,9]:

	O_1	O_2	O_3	O_4	O_5	O_6	....
S_1	W		E	R,W	R,W		
S_2				P	P		
S_3				R	R	R,W	
S_4	W		E	R,W	R,W		
⋮							

In this matrix, the S_i denote the subjects and O_i the objects. The entries $A(S_i,O_i)$ denote the access attributes, for example, R, W, E, and P (i.e., Read, Write, Execute, and Print, respectively).

In examining the matrix, we make the following observation:

(1) There are usually more objects than subjects. This is particularly evident in a data base environment. Thus, the number of columns in the matrix will be much greater than the number of rows.

(2) The matrix is sparse, especially when each subject has access to a relatively small number of objects in the data base system.

(3) In the same matrix two or more rows may be identical, indicating that two or more subjects have identical access attributes with respect to the same objects. As far as the access control mechanism is concerned, these subjects are alike (e.g., the rows identified by S_1 and S_4 in A are identical.).

(4) Similarly, two or more columns in the access matrix may be identical. When this happens, the objects may be accessed in the same manner.

(5) Many matrix entries are identical. For example, $A(S_1,O_4)$ and $A(S_1,O_5)$ are identical.

Because the access matrix is sparse and has many columns (i.e., many objects involved), attempts have been made to organize the matrix into manageable pieces for effective use. Consider the following two approaches. One is to organize for each subject a list of access attributes of all the objects accessible to the subject. In this approach, the subject-row approach, there is a list of access attributes specified for the objects which the subject has been authorized to access where as the inaccessible objects do not have access attributes associated with the list. Thus, the list is compact and subject-oriented. Examples of using the subject-row approach for managing access matrices in contemporary operating systems are the so called capability list systems [3,8]. On the other hand in the object-column approach, for every object in the system there is a list of subjects who have been given access with appropriate access attributes to the object. Obviously, subjects who have no access to an object do not have their access attributes included in the list for the object. Thus, the list is again compact. However, it is object-oriented. An example of using object-column approach for access matrices management in current operating systems is the so called access list systems [2,10]. In both the capability and access list systems, the aim is to implement the access matrix as depicted earlier without the redundant entries.

By organizing the access matrices into capability lists and linking the entries for the objects in one list with the same objects in another list, it is possible to have a capability list system whose listed objects constitute effectively an access list system. Thus, in this approach both subject-row and object-column are emphasized. Although the linkage requires additional cost and processing, this approach entertains considerable flexibility. An example of using this approach in data base systems can be found in the authority item systems [6,7].

The access matrix as implemented in the capability list, access list and authority item systems is well-suited to contain access control information about the subjects and objects individually and explicitly. By individually and explicitly we mean that before a subject S_i is granted or denied the access to an object, the access control mechanism simply checks the access attributes associated with the entry $A(S_i,O_i)$. There is no need for the access control mechanism to consult any other entry, say, $A(S_i,O_j)$ for access to the object O_j by S_i. In other words, access attributes associated with S_i and O_i are explicitly contained in $A(S_i,O_i)$. Furthermore, these and only these access attributes can cause the granting and denying of the subject's access to the object.

In a real data base system environment, access attributes are often implicitly defined as relations among objects and subjects. For instance, the specification "S_2 may not print O_5 together with O_4" is a relation between objects O_4 and O_5 over subject S_2. Similarly, the specification "O_2 can only be read by S_3 if it has been written by S_6" is a relation between subjects S_3 and S_6 over object O_2. These relations, which we shall call access control relations, are difficult if not impossible to represent in an access matrix. There are also more subtle access control relations. It is, therefore, the aim of this study to investigate some of the basic access control relations and to establish the rules which govern the accessing of objects by subjects on the basis of these relations. Particular emphasis will be on objects whose entities are real-world data items such as fields, records, files, and directories. We will refer to a data base system whose access control mechanism can enforce access control relations among objects a data secure system.

2. The Choice of a Graph-Theoretic Approach

As mentioned in the previous section, the access matrices are not adequate to represent the access control information of a data secure system. Instead, directed graphs will be used. There are several reasons for using directed graphs. First, objects and simple relations among objects can easily be represented by nodes and edges between nodes. Secondly, in the course of investigating simple access control relations, it is discovered that the order in which objects are being accessed is important. This access ordering of objects is readily representable by a path in a directed graph. Finally, directed graphs and mechanisms based on directed graphs have borne satisfactory results in some other areas such as computer system deadlocks [5]; we think that directed graphs may also be useful in the study of data secure systems.

2.1 Some Definitions and Restrictions

Some definitions of terms from graph theory are given here. A directed graph is a pair (N,E), where N is a nonempty set of nodes and E is a set of edges. An edge $e \in E$ is an ordered pair (a,b) such that a and b are nodes in N. The edge (a,b) is said to be directed from a to b . An isolated node is a node with no edges directed to or from it. A path is a sequence of at least two nodes (a,b,c,...,t,u), where (a,b), (b,c),..., and (t,u) are edges. A cycle is a path whose first and last node are the same.

A user is a group of one or more subjects such that they all have the same access attributes to the same objects. In referring to the matrix notation, we have, for example, that any two subjects S_1, S_2 are members of a user U (i.e., $S_1, S_2 \in U$) if and only if $A(S_1,O_i) = A(S_2,O_i)$ for i = 1, 2, ..., n, where n is the number of objects (columns) in matrix A. It is clear that each subject belongs to one and only one user and users are mutually disjoint. Furthermore, the subject-oriented systems become user-oriented systems.

The grouping of objects is similar to the grouping of subjects, i.e., identical columns in the matrix, for example, are grouped together. In a user-oriented system, the number of object groups for each user can be reduced even further. For the user U, two objects O_1, O_2 belong to the same object group G (i.e., $O_1, O_2 \in G$) if and only if $A(S,O1) = A(S,O_2)$, where S is any subject of U. We note that the grouping of objects is different for different users in a user-oriented system. Also, each object belongs to one and only one object group. In other words, if $O \in G_1$ and $O \in G_2$ then $G_1 = G_2$. In our discussion, the object groups are called data units.

We would like to impose a few restrictions at the beginning to facilitate the discussion of access control information. Generalizations will come at a later time. The first restriction is that users are not considered as objects. This does not mean that users need not be protected since the only protected entities in the system are objects. Its sole purpose is to ease the discussion. The second restriction is that instead of creating one large directed graph for all the access control information, a directed graph is created for each user. As we recall that the access control information of the data secure system constitute, at a given time, all the specifications regarding the types of access to objects that subjects have in the system. This information is too large to be used for subsequent discussion. Therefore, at a given time only the part of the access control information which deals with the data units that the user has will be depicted by a directed graph. Nevertheless, the set of directed graphs for all users of the system represents the access control information of the system. The directed graphs are called <u>user directed graphs</u> (UDG).

To distinguish one type of nodes from another, different shapes of nodes and edges are used in a user directed graph. A square node is used to represent the user while circular nodes are used for each data unit involved with this user. Data units which are not accessible to the user are not included. The collection of all accessible data units for a user is called the <u>user's logical data base</u>.

Each node will contain a certain amount of information. The square node contains only the name of the user. A circular node contains two kinds of information: the name of the data unit and the access rights (i.e., attributes) the user has to access that data unit. The name and attributes are separated by a slash (/). For example, if D1 is the name of a data unit for user U, then a node containing D1/A(S,O) for each $S \in U$ and every $O \in D1$ must be present. This is too redundant. We thus come to the third restriction that each data unit is represented by one and only one node in a user directed graph. Thus the name of a data unit will be used to denote either the data unit or the node representing the data unit.

Only two different types of edges are to be discussed here: the edges between the square node and circular nodes and edges between circular nodes. An edge represented by a single arrow directed from the square (user) node to a circular node indicates that the user is <u>requesting</u> an access to the data unit represented by that node. If the arrow is directed in the opposite direction, it indicates the <u>granting</u> of the desired access to the data unit for the suer. We term the former the <u>request edge</u> and the latter the <u>grant edge</u>. A grant edge cannot be deleted, while a request edge can be deleted. In fact, these are the only edges which do not represent access control relations.

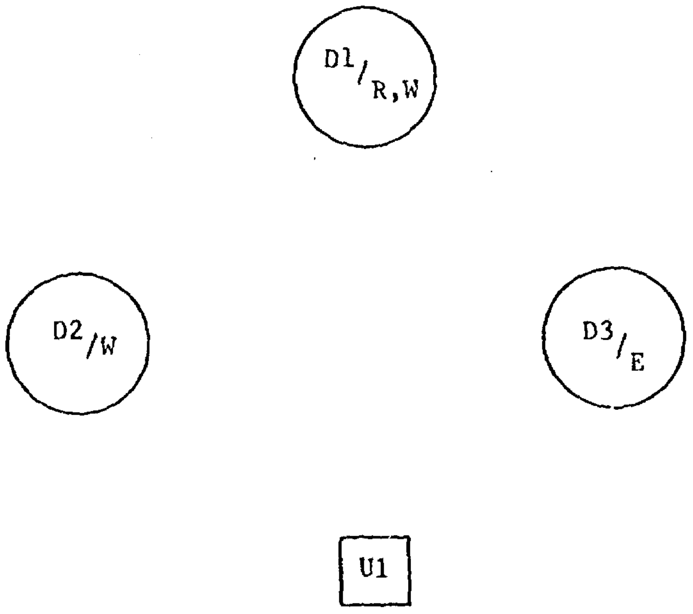
2.2 Built-in Access Control Relations

Directed edges between circular nodes represent access control relations between data units. The discussion of access control relations between data units will occupy a substantial part of this study in the following sections. Here we simply illustrate the absence and presence of access control relations between data units by way of directed graphs.

Example 1. Consider the first and fourth rows of the access matrix A presented earlier. Since these two rows have the same access attributes with respect to the objects, the two subjects S_1 and S_4 belong to the same user U1. There are exactly four accessible objects to the user, namely, O_1, O_3, O_4, and O_5. However, O_4 and O_5 have the same attributes with respect to the subjects, they can be grouped into one data unit. In summary, we have

$$U1 = \{S_1, S_4\},$$
$$D1 = \{O_4, O_5\},$$
$$D2 = \{O_1\},$$
$$D3 = \{O_3\}.$$

In the user directed graph, we have

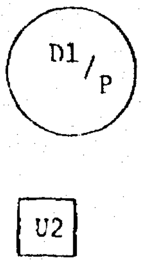


ERIC

Example 2. Consider the second row of the access matrix A. We have the following

$U2 = \{S_2\}$,

$D1 = \{O_4, O_5\}$.

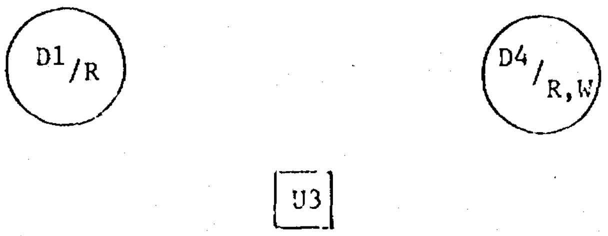


Example 3. Consider the third row of the access matrix A. We have

$U3 = \{S_3\}$,

$D1 = \{O_4, O_5\}$,

$D4 = \{O_6\}$.

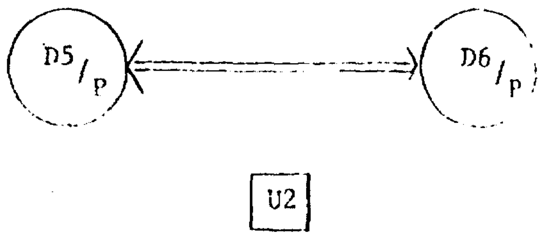


The above examples show that the access matrix A can be easily represented by three directed graphs. In addition, as we shall see, directed graphs can facilitate the representation of access control relations which are difficult to be included in the access matrix. For instance, let us impose the access control relation "S_2 may not print O_5 together with O_4" onto one of the above examples, namely, example 2 and represent the relation with a bi-directional edge. The directed graph then takes the following form for

$U2 = \{S_2\}$,

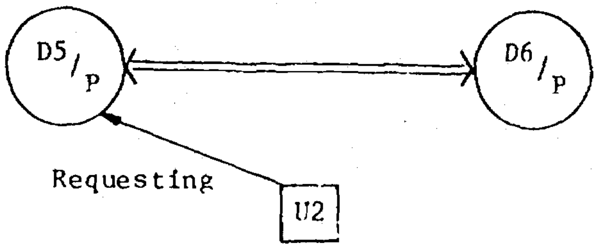
$D5 = \{O_4\}$,

$D6 = \{O_5\}$.

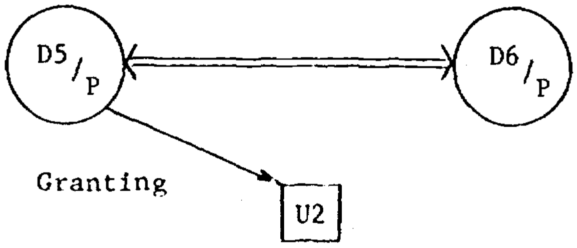


Now several observations are worth stating. First, the access control relations among data units, as depicted by the bi-directional edge in the above example, are built-in relations. Their presence among data units may cause data units to be used for more elaborate access control purpose. Secondly, at the very beginning all nodes in a user directed graph are isolated nodes, except those connected by edges representing built-in access control relations. Third, the granting of data units to a user is deemed "permanent". Consequently, grant edges cannot be deleted for a certain duration of time. In other words, the presence of the grant edges serves as a reminder that access has been given to certain data units. Without this reminder the access control mechanism cannot prevent a user from circumventing the access control relation among data units by making access requests at different times on the related data without regarding to the built-in restrictions. For instance, in referring to the user directed graph given above, we may have the following three cases.

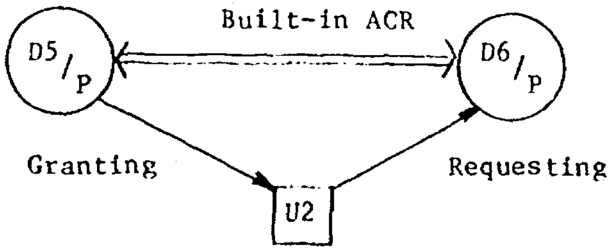
Case 1. The user requests the printing of data unit D5.



Case 2. The user's request is granted.



Case 3. The user U2 requests the printing of data unit D6.



Because the grant edge (from D5 to U2) is still there, the access control mechanism can determine whether access to D6 should be granted on the basis of the built-in access control relation between D5 and D6. If the grant edge is not "permanent", user U2 will be able to print both D5 and D6 by submitting one job for printing D5 and then another job for printing D6, in spite of the given access control relation.

Throughout this report, we will use the following notations:

U, U1, U2, U3, ...	for representing and naming users
D, D1, D2, D3, ...	for data units
A(x)	for access attributes of data unit x.

3. Context Protection

Intuitively, we say that a data unit D2 is the context of another data unit D1 if the access attributes of D1 is affected by the accessing of D2. For example, consider data units D3 and D4 such that D4 cannot be printed if D3 has been printed. D3 is the context of D4. From now on, if D2 is the context of D1, we call D1 the text of D2.

3.1 Three Cases of Context Protection

Context protection means that the same data unit may have to be accessed differently in different contexts. At least three cases can be identified.

Case 1: A data unit can be accessed based on certain access attributes such as write and print if its context has not been accessed by the user. However, it can not be accessed at all when its context has been accessed by the same user. This case is a simple case of context protection.

Case 2: A data unit has different access attributes associated with different contexts. For example, a personnel officer may have the rights (i.e., proper access attributes) to get a list of the names of all employees and a separate list of the salaries of all employees. Furthermore, the officer is directed from time to time to change the salaries of certain employees. Nevertheless, he is not allowed to list the salaries while names are being listed. In this example, the data unit of employee names is the context of the data unit of employee salaries. As far as the data unit of names is concerned, it involves no context protection because this data unit is not in the context of any data unit. The access to the data unit of names is solely determined by the access attributes of the data unit. On the other hand, the data unit of salaries can only be listed when its context is not involved and can be changed when its context is indeed involved. In other words, the access to the data unit of salaries depends upon whether the data unit of names has been accessed or not. Such is a more advanced case of context protection.

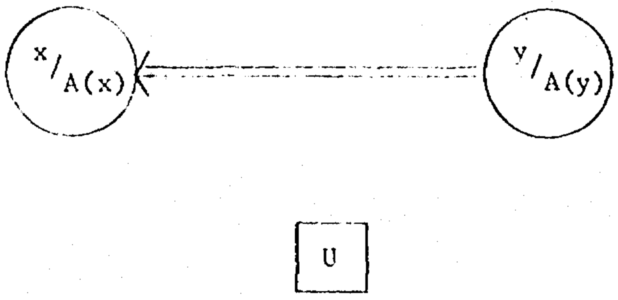
Case 3: If a data unit is the text of more than one context and for each context there is a set of different access attributes associated with the text, then this case is a generalization of case 2. For example, suppose the data unit D1 is the text of data units D2 and D3. Whenever the data unit D2 is accessed, the access attributes of D1, i.e., A(D1), will become ar_1. Whenever the data unit D3 is accessed, A(D1) will become ar_2. These two sets of access attributes for D1, namely, ar_1 and ar_2, are usually different.

Thus, by context protection we mean also the control of access to data units involving the above three cases.

3.2 Access Control Relations for Context Protection

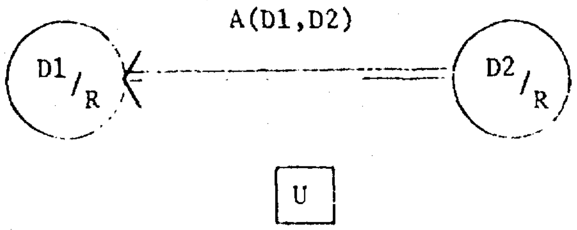
As we have discussed earlier, there are various kinds of access control relations in a data secure system. Essentially, these relations define the manner in which access to the related data units should take place and the access attributes with which the data units should be controlled. In particular, we are interested in those access control relations which have an impact on context protection. It is hoped that by identifying these access control relations, effective access control mechanisms can be developed for the data secure system to enforce the context protection of data units. The following three relations are proposed. It should be obvious to the reader that these three access control relations have resulted from the study of the three cases in the previous section.

An access-order relation AO(x,y) with respect to a user U is a relation between two data units x and y in a user directed graph such that y cannot be accessed in any manner by the user U when an access to x has been granted to the same user. Graphically, this relation is represented by

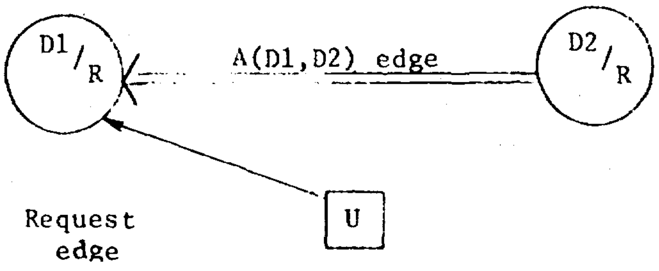


where A(x) and A(y) are access attributes of x and y, respectively.

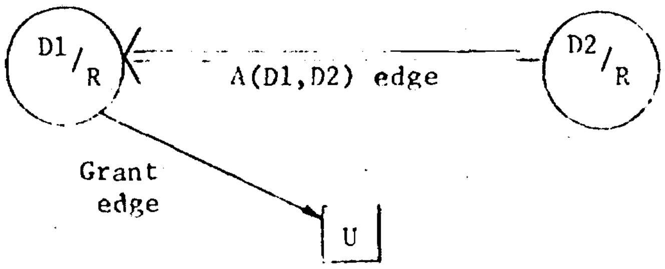
The access-order relation can be used to characterize the access requirements that, for example, if the user has access to this record for reading then he cannot access the other record for reading. Let the user be U, this record be data unit D1, the other record be data unit D2, and the access attribute "reading" be R, we have the following user directed graph.



When the user makes a request to read D1, the user directed graph becomes



The request is granted. The user directed graph is now as follows



Sometime later, the user requests the reading of the other record. This is reflected in the user directed graph as follows

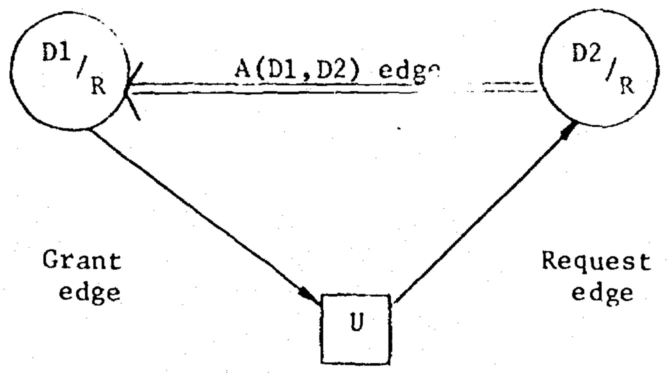
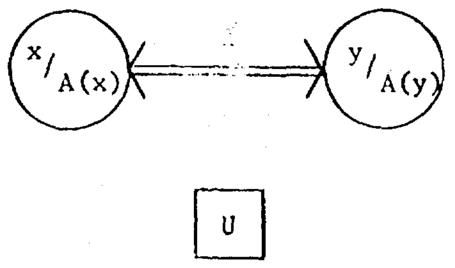


Figure 1

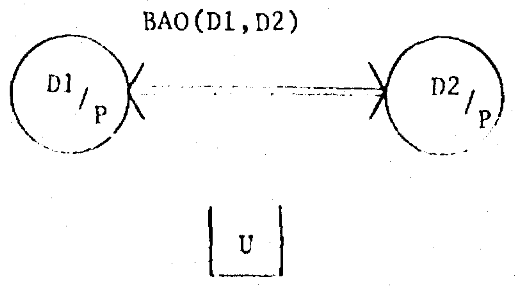
The question is whether the new request is to be granted. Based on the access requirement, this request should be denied since it was understood at the outset that D2 cannot be read if D1 has been read by U.

A <u>bi-directional access-order relation</u> BAO(x,y) with respect to a user U is a relation such that only one of data units x and y can be accessed by the user. Graphically, it is represented by



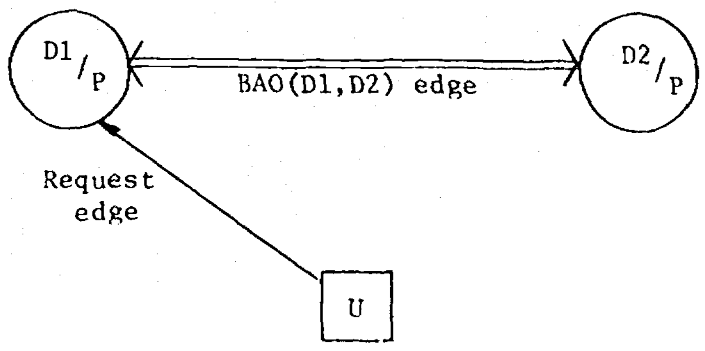
We observe that BAO(x,y) implies AO(x,y) and AO(y,x). Furthermore, BAO(x,y) = BAO(y,x).

The bi-directional access-order relation can be used to characterize the access requirements that the user can print the employee names or the employee salaries but not both the names and salaries. Let the user be U, the set of employee names be the data unit D1, the set of employee salaries be the data unit D2, and the access attribute "print" be P. Thus the user directed graph is as follows.

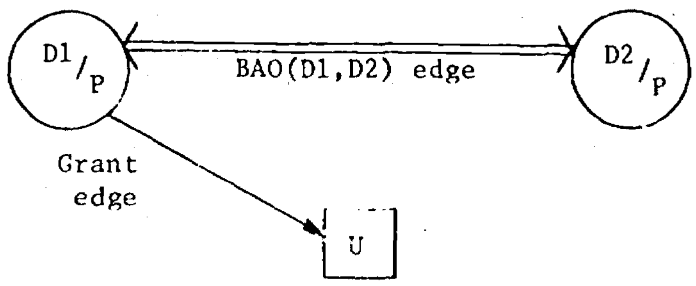


Now a scenario of requesting access to data units by the user and granting access to data units for the user will be depicted as follows.

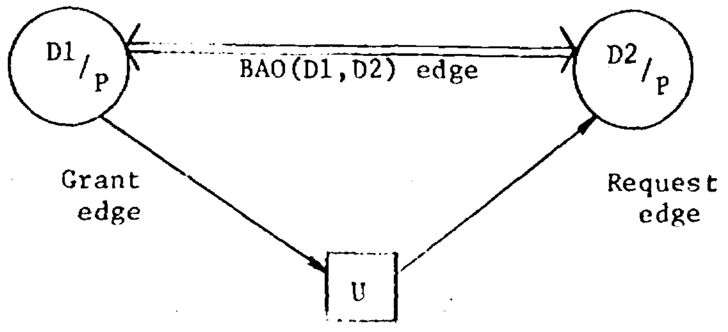
The user is requesting a list of data unit D1.



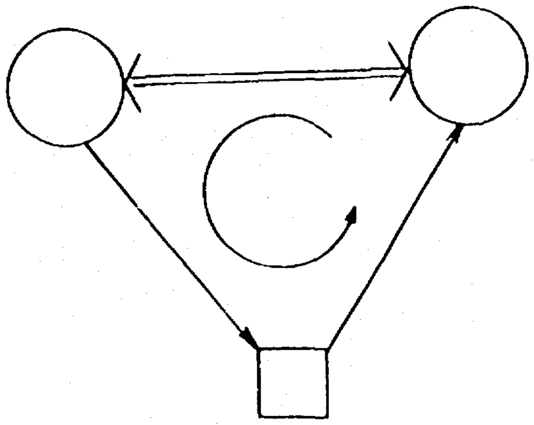
The request is granted as reflected in the following user directed graph.



Although the first request has been granted, the user is making a request for listing the data unit D2 as follows.



Can the second request be granted? Of course, not. Let us look at the user directed graph again without annotation.



We note that the directions of the edges form a cycle. We further note that for the case involving the access-order relation in Figure 1, there is also a cycle. The cycles in user directed graphs are intended to reveal those requests whose granting may render access control relations invalid.

3.3 Violation of Access Control Relations

An access control relation between data units is violated if the granting of a request for one of the data units makes the definition of the relation false. A violation of protection in a user directed graph is therefore defined to mean that one or more access control relations (so far only AO and BAO) in the user directed graph are violated due to the granting of a user request. With these definitions, we can prove the following theorem.

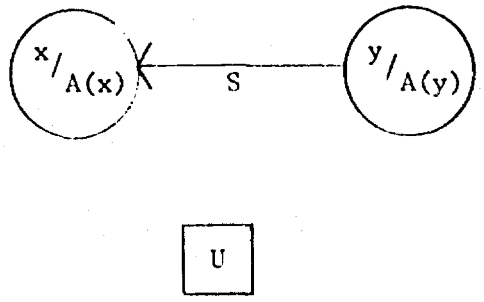
A violation exists if and only if there is a cycle in a user directed graph such taht

1. one node in the cycle is the user (square) node,
2. each circular node in the cycle is either granted or requested, and
3. the edges, except the grant and request edges, represent AO or BAO relations.

3.4 The Context Relation

Although access control relations such as AO and BAO are simple and useful, they are not as powerful as desired. They are unable to enforce the last two cases of context protection. Thus, a more powerful relation must be introduced. The new relation must have the ability to pass access control information from the context to the associated text in order to determine the new attributes of the text, when the context is accessed. This new relation is called a <u>context relation</u>.

A <u>context relation</u> C(x,y) is a relation between two data units x and y such that y is the context of x. The node x is called the <u>text node</u> (TN). The node y is called the <u>context node</u> (CN). They are regular circular nodes. These two nodes are connected by a directed edge from CN to TN. In addition, there is associated with the edge a set S of access attributes for access attributes in A(y). Graphically, the relation is represented by



where $S = \{S(a) \mid a \in A(y)\}$. For example, if $A(y) = \{a_1, a_2\}$, then $S = \{S(a_1), S(a_2)\}$, where $S(a_1)$ and $S(a_2)$ are sets of access attributes associated with the edge. When an access to y is granted with access attribute a_i to a user, the new access attributes of x will be determined jointly by A(x) and $S(a_i)$. The ways of determining the new access attributes of x based on the attributes of x (i.e., A(x)) and the imposed attributes (i.e., S) will be given later in this section. On the other hand, if the data unit y has never been accessed, the imposed access attributes have no effect on the subsequent access to data unit x. Intuitively, A(x) and A(y) are used to determine access requirements to x and y, respectively, when no context protection is required. However, if there is any need of context protection, a set S of access requirements is created which exercises control over the access requirements of the text, in this case, x. This control is related to every access control requirement of the context, y. In other words, for each access requirement a_i of A(y) there is a related set of access requirements $S(a_i)$ of S. the $S(a_i)$ will be used in conjunction with A(x) to form new access control requirements for accessing x. With these new access control requirements, the protection of x in the context of y can be assured.

We note that the edge directed from a context node to a text node is the same as the request/grant edge. We do not care to distinguish them because we always know the difference with the presence of S. When there are two or more context nodes in a user directed graph, the S's may be subscripted by the names of these context nodes. For example, if y and z are two CNs of a UDG, then we may have S_y and S_z, each of which is associated with an edge leading from the respective CNs.

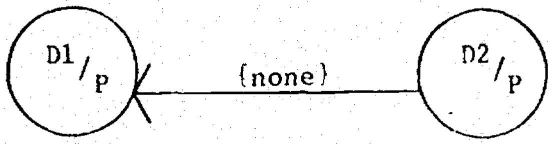
Let us consider a simple example of two data units.

D1 can be printed (by a user).

D2 can be printed.

D1 cannot be printed if D2 has been printed.

From the above specifications, we can construct the relation C(D1,D2) with A(D1) = A(D2) = {P} (P for print) and S = {S(P)} = {none}, where "none" means no access at all. Graphically,



Now the new access attribute of D1 will be "none" if D2 has been printed since the rule (to be discussed in sequel) says that the existing access attribute P of D1 must be replaced by the new attribute "none" for accessing D1.

Two things are made clear by the above example. First, everything that can be done by AO or BAO relations can be done by context relations. Secondly, there must be rules to determine the new access attributes of a text node based on its old access attributes, access attributes of one of its context nodes, and the access attributes associated with the edge leading to the text node. Indeed, two rules will be proposed which can determine the new access attributes of a text node and govern the granting and denying of an access to a context node. The granting and denying of an access to a text node require no rules and are straightforward.

Before introducing the rules, we introduce the following conventions. The access attributes given in S are represented by lower case letters, e.g., r (read), w (write), p (print), etc. This is to distinguish them from the access attributes of a circular node which are in upper case since these attributes play different roles in the rules. Another point to be mentioned is that the set A(x), where x is a data unit, usually consists of more than one access attributes. In order to distinguish which access attribute is granted to a user, the letter representing the granted attribute is flagged. Although a grant edge is needed each time a new attribute of the data unit is granted to the user, multiple grant edges for the same data unit are not necessary. Since granted attributes are flagged, we can simply identify the flags. Thus, a single grant edge for the data unit will suffice.

The operations in the rules are set operations because A(x), S, and A(y) are considered as sets of access attributes. In the rules, "-", "$\cap$", and "$\leftarrow$" are set difference, intersection, and assignment, respectively. Formally, if A and B are two sets, then

$$A - B = \{x | x \in A \text{ and } x \notin B\},$$
$$A \cap B = \{x | x \in A \text{ and } x \in B\}, \text{ and}$$
$$A \leftarrow B = \{x | x \in B\},$$

In our application, an access attribute, whether it is in a capital letter, small letter, or flagged letter,is used as the same attribute. Thus, r, R, and $\bar{R}$ are treated in the set operations as the same element. For example, if A = $\{\bar{R},P\}$ and B = $\{r\}$, then A - B = $\{P\}$. That is, $\bar{R}$ and r are considered as the same element in the set operation. Furhtermore, small letters override the corresponding capital letters in the set intersection operation. For instance, if A = $\{W,P\}$ and B = $\{w\}$, then $A\cap B = \{w\}$. Finally, all flags are retained in the assignment and intersection operations in order to keep track of the granted accesses. Suppose A = $\{\bar{W},P\}$ and B = $\{w,e\}$. Then

$$A \leftarrow B = \{\bar{w},e\}, \text{ and}$$
$$A \cap B = \{\bar{w}\}.$$

Now the rules are given below. In addition to the notations, CN, A(CN), TN, A(TN), and S as defined previously, we use the symbol "ar" to mean the access attribute of CN for which the user has just made an access request. Obviously, ar is a member of A(CN).

Detection Rule 1: If every member of A(TN) is a capital letter and there exists a flagged member in A(TN) - S(ar), then the granting of the user's request will result in a <u>violation</u> of context protection. If at least one member of A(TN) is in lower case or no member of A(TN) - S(ar) is flagged, there will be no violation. In this case, we replace A(TN) with A(TN) $\leftarrow$ S(ar).

Detection Rule 2: If every member of A(TN) is a lower case letter and there exists a flagged member in A(TN) - S(ar), then the granting of the user's request will result in a <u>violation</u> of context protection. If at least one member of A(TN) is in upper case or no member of A(TN) - S(ar) is flagged, there will be no violation. In this case, we replace A(TN) by A(TN) $\cap$ S(ar).

The application of the rules whenever there is a user request is facilitated by the following <u>resolution algorithm</u>.

(1) Determine the access attribute, ar, of the data unit x for which the user has requested.

(2) If ar $\notin$ A(x), deny the request. Return.

(3) Since ar ε A(x), determine if there is a non-empty S_x. If none, go to step (6).

(4) Since $S_x \neq \emptyset$, apply either Rule 1 or Rule 2. In this application, x is obviously a CN.

(5) If there is a violation, deny the request. Otherwise, flag ar in A(x). Return.

(6) Invoke the theorem.

(7) If there is a violation, deny the request. Otherwise, flag ar in A(x). Return.

With the availability of the two rules, the theorem and the algorithm, the access control mechanism of a data secure system can grant and deny requests and reveal violations. However, there are two types of violations. A violation resulted in step (5) is a severer case of violation than the one resulted in step (7). The former may damage the context relations between two data units; the latter can only sever the AO or BAO relations. Thus, the algorithm can resolve the difference of these two types of violation. Both rules are designed to detect potential violations of context protection among data units. Because the detection of a violation of AO or BAO relations does not require elaborate rules as in the case of context relation, a theorem for the violation detection is provided. With the aid of the resolution algorithm, the access control mechanism of a data secure system enforces expeditiously and properly controlled access to data units of the data base for the user.

4. Application of Context Protection to Job Processing

Sample 1: Enforcement of a more advanced case of context protection.

User data base: user name U; data units D1 and D2.

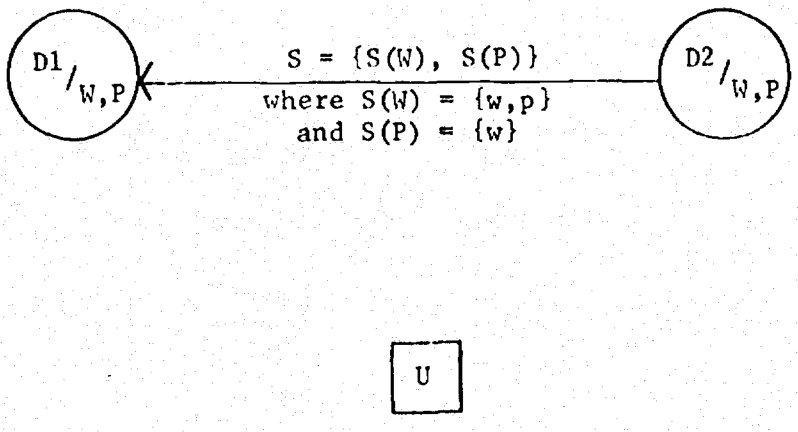
Access attributes specifications:

D1 can be written or printed, i.e., W and P ε A(D1);

D2 can be written or printed, i.e., W and P ε A(D2); and

D1 cannot be printed if D2 has been printed, i.e., S.

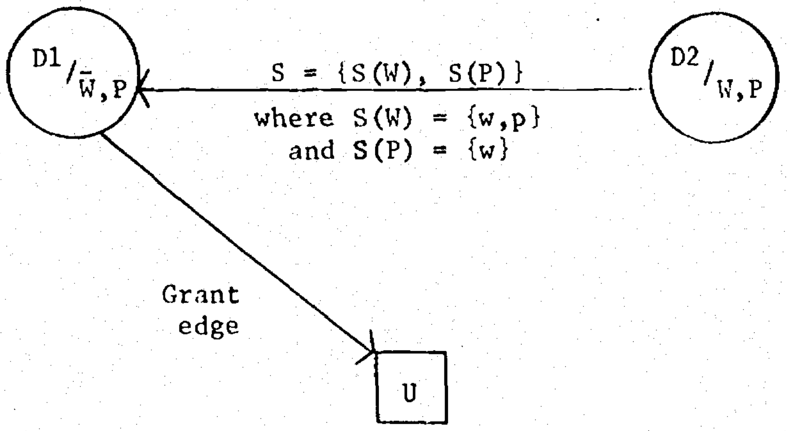
User directed graph (UDG):



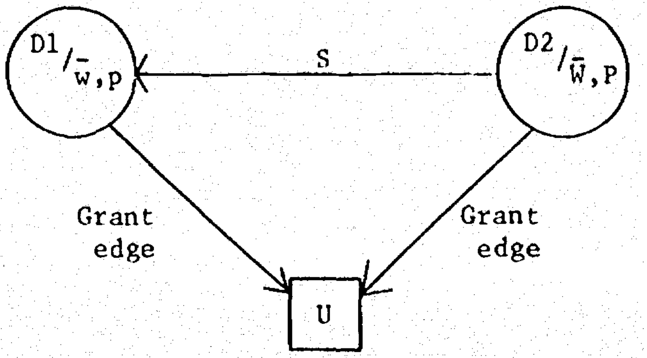
Jobs: (For the ease of discussion, we suppose that only one job is ever submitted by the user U. This job will be one of the following four jobs.)

Job 1: Write D1
Write D2

According to the access control requirements, both requests of the job can be granted. The first request is granted because the proper access attribute W is in A(D1) and the request (edge) will not result in a cycle. Thus by the theorem, there will be no violation. After the first request is granted, the UDG looks like as follows.



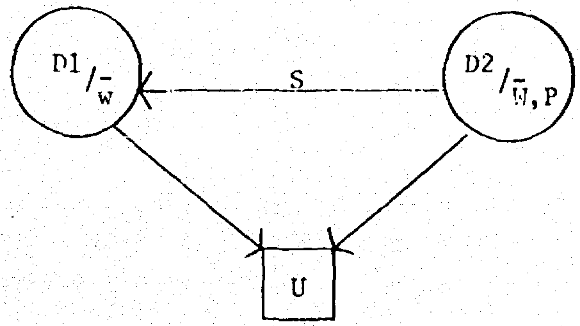
We note that the access attribute W of D1 is flagged indicating that a write request has been granted. The second request is also granted due to the fact that it is cleared by Detection Rule 1 with $A(D1) = \{\bar{W},P\}$, $S(W) = \{w,p\}$, and $A(D1) - S(W) = \emptyset$. The final graph is therefore



We note that the access attributes of D1 are replaced by $\{\bar{w},p\}$, i.e., A(D1) is replaced by $A(D1) \leftarrow S(W)$ as dictated in Rule 1. Lower case letters in A(D1) indicate that certain context protection has been facilitated.

Job 2: Write D1
Print D2

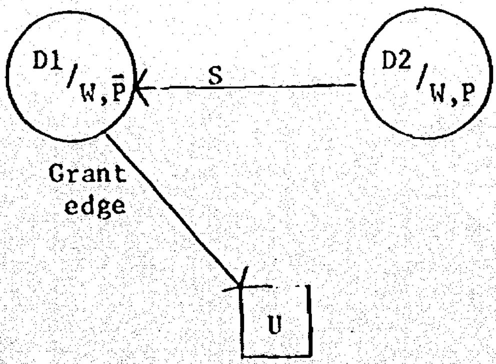
Again, both requests in the job can be granted according to the specifications. The second request is granted because the Detection Rule 1 has cleared the request with $A(D1) = \{\bar{W},P\}$, $S(P) = \{w\}$, $A(D1) - S(P) = \{P\}$, and $A(D1) \leftarrow S(P) = \{\bar{w}\}$. The final UDG is



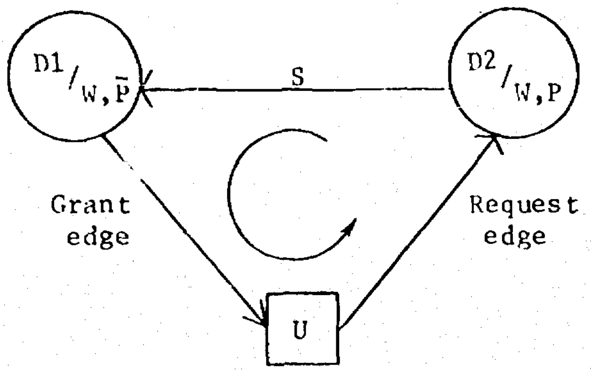
We note that S(P) replaces A(D1) as dictated by Rule 1 during the granting of the request.

Job 3: Print D1
Print D2

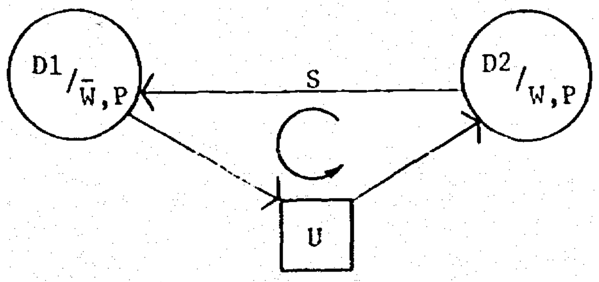
In this case, only the first request of the job can be granted. The second one is not granted because for D2 to be printed D1 must not have been printed. More specifically, the UDG after the first request is granted is



The Resolution Algorithm upon receiving the second request will invoke Detection Rule 1. Since $A(D1) = \{W,\bar{P}\}$, $S(P) = \{w\}$, and $A(D1) - S(P) = \{\bar{P}\}$, there is a violation due to the flagged P. The second request will therefore be denied.
Graphically, a cycle is also formed in this case. That is,

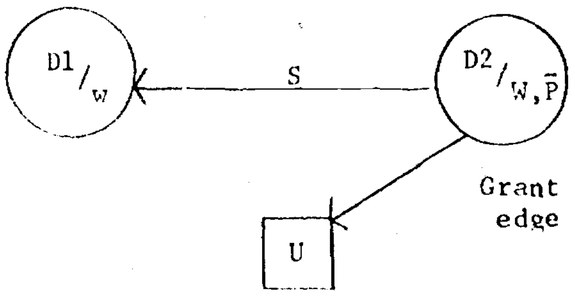


Nevertheless, a cycle does not necsssarily mean a potential violation when the relation involved is a context relation. For example, a cycle existed when the second request was made in job 2. That request was indeed granted. On the other hand, a cycle involving either AO or BAO relations always implies a potential violation as stated in the Theorem.



Job 4: Print D2
       Print D1

The first request can be granted because $A(D1) = \{W,P\}$, $S(P) = \{w\}$, and $A(D1) - S(P) = \{P\}$. By Detection Rule 1, there is no potential violation. Thus the request is granted and A(D1) is replaced by {w}, i.e., $A(D1) \leftarrow S(P)$.



It is now clear that the second request of the job for printing data unit D1 cannot be granted because $P \notin A(D1)$.

Sample 2: Enforcement of a more elaborate case of context protection involving several contexts.

User data base: user name U; data units D1, D2, and D3.

Access attributes specifications:

D1 can be written or printed;

D2 can be written or printed;

D3 can be executed;

D1 cannot be executed if D2 has either been printed or written; and

D1 can be executed only if D3 has been executed.

It is worth noting that there are five access attributes specifications in this data base. The first three specifications dealing with data units individually are therefore free from context access control considerations. The last two specifications are context-dependent. Furthermore, we note that the data unit D1 is the text node of the context nodes D2 and D3. More specifically, the above specifications can be represented symbolically as follows:

$A(D1) = \{W,P\}$,

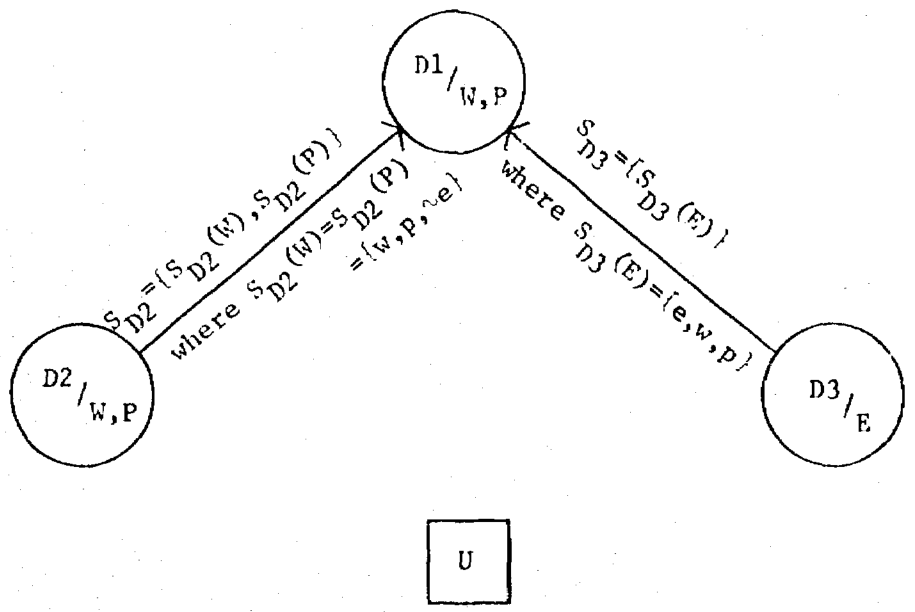
$A(D2) = \{W,P\}$,

$A(D3) = \{E\}$,

$S_{D2} = \{S_{D2}(W), S_{D2}(P)\}$, where $S_{D2}(W) = \{w,p,\sim e\}$ and $S_{D2}(P) = \{w,p,\sim e\}$,

$S_{D3} = \{S_{D3}(E)\} = \{\{e,w,p\}\}$.

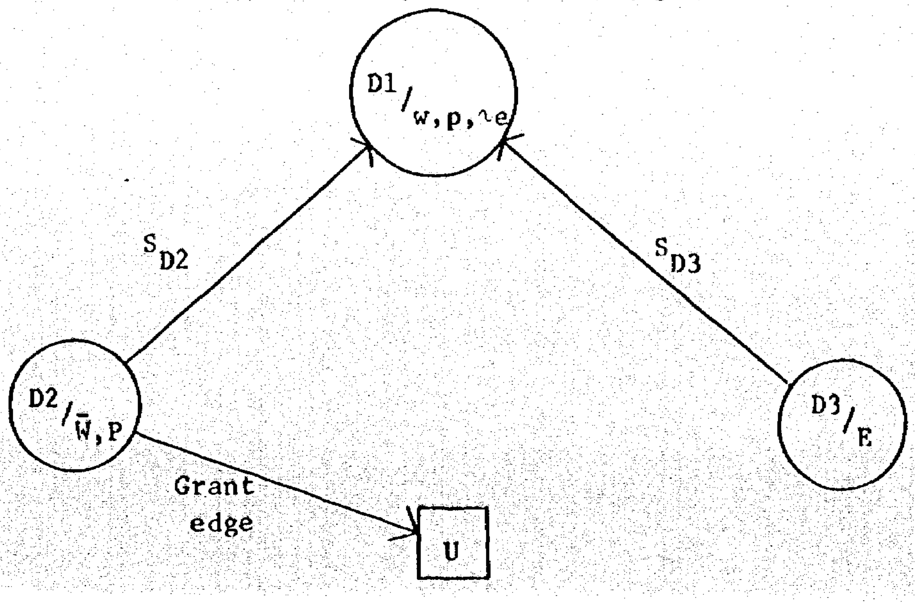
User directed graph:



Jobs: (The assumption that only one job is ever submitted by the user U for processing is intended to simplify the discussion. The one job will be either of the following two.)

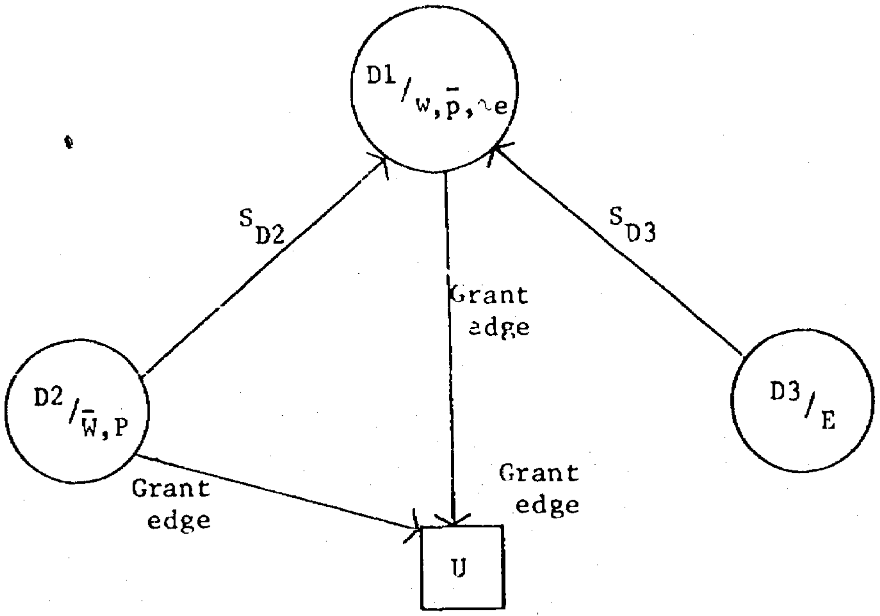
Job 1: Write D2
Print D1
Execute D3

The first two requests of the job can be granted by referring to A(D1) and A(D2). The last one can also be granted by referring to A(D1) and A(D3). Let us follow through the scenario of the first two requests. D2 is granted by Detection Rule 1, since $A(D1) = \{W,P\}$, $S_{D2}(W) = \{w,p,\sim e\}$, and $A(D1)-S_{D2}(W) = \emptyset$. There is no violation. Thus, the UDG becomes

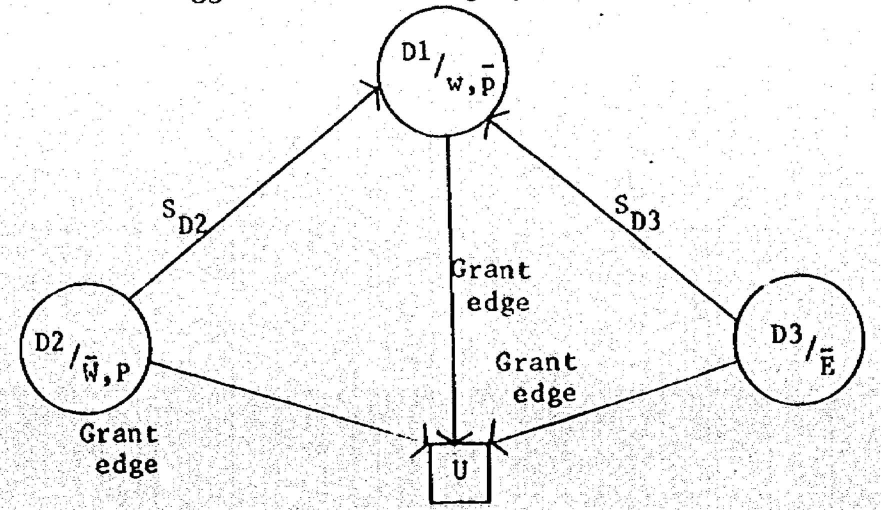


ERIC Full Text Provided by ERIC

The granting of the first request causes the $W \in A(D2)$ to be flagged. Also $A(D1)$ is replaced by $A(D1) \leftarrow S_{D2}(W)$. The second request is also granted because $P \in A(D1)$ and no S is associated with D1. By invoking the Theorem, no cycle is detected. Therefore,

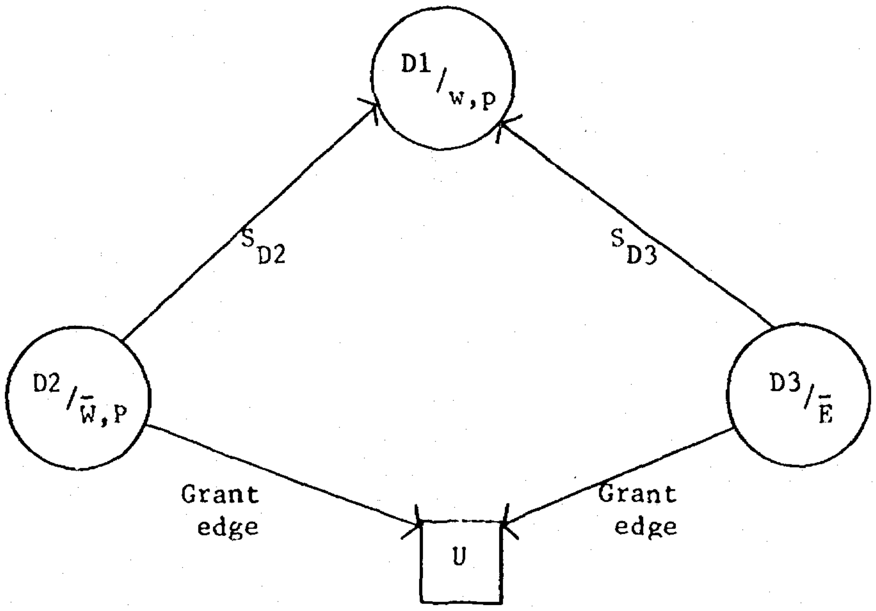


Since the second request for printing is granted, the p in $A(D1)$ is flagged. When the third request is processed, Detection Rule 2 is invoked. We have $A(D1) = \{w,\bar{p},\sim e\}$, $S_{D3}(E) = \{e,w,p\}$, and $A(D1)-S_{D3}(E) = \emptyset$. Thus, every member of $A(D1)$ is in lower case and there exists no flagged member of $A(D1)-S_{D3}(E)$ which implies no potential violations. The request must therefore be granted and $A(D1)$ is replaced by $A(D1) \cap S_{D3}(E) = \{w,\bar{p},\sim e\} \cap \{e,w,p\} = \{w,\bar{p}\}$ where $\{\sim e\} \cap \{e\} = \emptyset$. Furthermore, the access attribute E in $A(D3)$ is flagged. The final graph is



Job 2: Write D2
Execute D3
Execute D1

The first two requests can be granted, but the last one must be denied due to context-depdependent access control requirements. After D2 has been written, the UDG for this job is the same as the first graph of Job 1. When the second request is processed, Detection Rule 2 is invoked. We have A(D1) = $\{w,p,\sim e\}$, $S_{D3}(E) = \{e,w,p\}$, and $A(D1) - S_{D3}(E) = \emptyset$. Since the members of $A(D1) - S_{D3}(E)$ are not flagged, there is no potential violation and the second request is granted. Furthermore, A(D1) is replaced by $A(D1) \cap S_{D3}(E) = \{w,p\}$. The UDG becomes



Obviously the third request cannot be granted since by now A(D1) = $\{w,p\}$ and $e \notin A(D1)$. Without the proper access attribute in A(D1), there is no way that the data unit D1 can be accessed.

5. Summary

In this report, we first pointed out the inadequacies of the access matrix approach to represent the access control information in a data secure system. Several existing attempts to organize the matrix into manageable pieces for effective use were then discussed. They are well-suited to contain access control information about the subjects and objects individually and explicitly. However, in a real data base system environment access attributes are often implicitly defined as relations among objects and subjects. To cope with this problem, a graph-theoretic approach was therefore chosen. In this approach, access control relations among data units can easily be represented by edges between nodes, which represent the involved data units. Those relations which are involved with the discussion of context protection are named Access-order relation, Bi-directional Access-order relation, and Context relation. These three relations, together with the Theorem on violation, two Detection Rules, and the Resolution Algorithm can be used to enforce the desired context protection as depicted by the three cases of context protection. Applications of context protection to job processing were illustrated at the end.

References

1. Bergart, J.G., Denicoff, M. and Hsiao, D.K., "An Annotated and Cross-Referenced Bibliography on Computer Security and Access Control in Computer Systems," Technical Report (OSU-CISRC-TR-72-12). The Computer and Information Science Research Center, The Ohio State University, November 1972.

2. Daley, R.C. and Newmann, P.G., "A General-Purpose File System for Secondary Storage," Proc. AFIPS 1965, FJCC, 27, 213-229.

3. Dennis, J.B. and Van Horn, E.C. "Programming Semantics for Multiprogrammed Computations," CACM, 9,3, (March 1966), 143-155.

4. Graham, S.G. and Denning, P.J. "Protection-Principles and Practice," Proc. AFIPS 1972, SJCC, 40, 417-429.

5. Holt, R.C., "Some Deadlock Properties of Computer Systems," Computing Surveys, 4, 3, (September 1972), 179-196; "On Deadlock in Computer systems," Ph.D. Dissertation, Cornell University, January 1971.

6. Hsiao, D.K., "A File System for a Problem Solving Facility," Ph.D. Dissertation, University of Pennsylvania, (May 1968). Also available from NTIS as report AD671826 and from the Moore School as technical report (68-33).

7. Hsiao, D.K., "Access Control in an On-line File System," File Organization: Selected Paper from FILE 68-An I. A. G. Conference, Swets and Zeitinger N.V., Amsterdam, 1969, 246-257.

8. Lampson, B.W., "Dynamic Protection Structures," Proc. AFIPS 1969, FJCC, 35, 27-38.

9. Lampson, B.W., "Scheduling and Protection in an Interactive Multiprocessor System," Ph.D. Dissertation, University of California at Berkeley, (March 1967).

10. Organick, E.I., The Multics System: An Examination of Its Structure, The MIT Press, 1972, 127-186 (Chapter IV).

COMPUTER &
INFORMATION
SCIENCE
RESEARCH CENTER