A Linear Time Algorithm for Deciding Subject Security

R. J. LIPTON AND L. SNYDER

Yale University, New Haven, Connecticut

ABSTRACT A particular protection mechanism from the protection literature—the take and grant system—is presented For this particular mechanism, it is shown that the safety problem can be solved in linear time Moreover the security policies that this mechanism can enforce are characterized

KEY WORDS AND PHRASES protection, security, take and grant, subject, object, linear time, transitive closure, operating system

CR CATEGORIES: 4 30, 4 31, 5 24

1. *Introduction*

The theoretical analysis of systems for protecting the security of information should be of interest to the practitioner as well as the theoretician. The practitioner must convince users that the integrity of their programs and files is maintained; i.e. he must convince them that the operating system and its mechanisms will correctly protect these programs and files Vague or informal arguments are unacceptable since they are often wrong. Indeed the folklore is replete with stories of "secure" systems being compromised in a matter of hours.

A primary reason for the abundance of these incidents is that even a small set of apparently simple protection primitives can often lead to complex systems that can be exploited, and therefore compromised, by some adversary. But it is precisely this fact, simple primitives with complex behavior, that lures the theoretician. Our purpose here is to present a concrete example of a protection system and then to completely analyze its behavior.

Our motivation for doing this analysis is twofold. The protection system that we study is *not* one we invented, rather it appears, for example, in Cohen [1] Moreover it is closely related to systems studied in Denning and Graham [2] and Jones [4]. This point is most important, for the space of possible protection systems is exceedingly rich and it is trivial to think up arbitrary systems to study. We are interested not in arbitrary systems, but in systems that have practical application

The above motivation is necessary but not sufficient for us to establish that these questions should interest the theoretician. Our second reason for studying these problems is that in a natural way they can be viewed as "generalizations of transitive closure." Informally, our model is:

Given: A directed labeled graph G and a set of rewriting rules $\mathcal{R}$.

Determine: Whether or not there is a sequence of graphs $G_1, G_2, \ldots, G_n$ such that $G = G_1$, G_n has property X and G_{i+1} follows from G_i by some rule in $\mathcal{R}$.

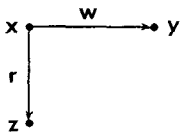
This work was supported in part by the Office of Naval Research under Grant N00014-75-C-0752 and in part by the National Science Foundation under Grant DCR74-24193

Authors' address Department of Computer Science, Yale University, 10 Hillhouse Avenue, New Haven, CT 06520

Here property X encodes that there is a protection violation in G_n. Our goal then is to determine whether or not such a G_n can be reached, i.e. to determine if a protection violation is possible.

Property X is frequently stated as follows: X there is an edge from vertex p to q with label α. This property looks very much like a transitive closure question. Indeed if the rules $\mathcal{R}$ only allowed the addition of arcs, then these problems would be easily solved by known methods. They are not so simple. The rules of interest to those in protection, and the particular rules we study, allow new vertices to be added. This simple change of allowing graphs to "grow new vertices" makes the problem challenging. Indeed the particular model we study is no longer even obviously decidable.

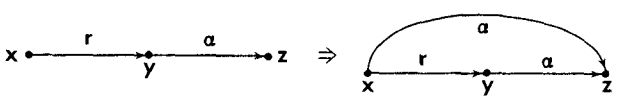
Let us now make the above concrete by introducing the particular protection system we study. We consider directed graphs whose arcs are labeled with an r or a w or a c and have no self loops Although we manipulate these graphs as formal objects, it is helpful to keep in mind the following informal semantics: A vertex corresponds to a *user*, r = *read*, w = *write*, c = *call*. If there is a directed arc from x to y with label r (respectively w, c), then x can *read* y (respectively *write*, *call*). For example, in the graph



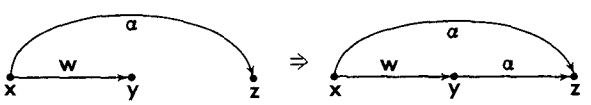
x can *write* y, x can *read* z, but y cannot *write* z since this edge is missing. More formally, a protection graph is a finite directed loop-free graph with each arc labeled by a nonempty subset of $\{r, w, c\}$. We interpret the case where an arc is labeled with other than a single element to mean that multiple "rights" are allowed.

This protection model, called the *take and grant system*, is now completed by presentation of five rewriting rules.

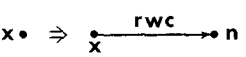
1. *Take*: Let x, y, and z be three distinct vertices in a protection graph, and let there be an arc from x to y with label γ such that $r \in \gamma$ and an arc from y to z with some label $\alpha \subseteq \{r, w, c\}$. The take rule allows one to add the arc from x to z with label α, yielding a new graph G' Intuitively x *takes* the ability to do α to z from y. We represent[1] this rule by



2. *Grant*: Let x, y, and z be distinct vertices in a protection graph G, and let there be an arc from x to y with label γ such that $w \in \gamma$ and an arc from x to z with label $\gamma \subseteq \{r, w, c\}$. The grant rule allows one to add an arc from y to z with label α, yielding a new graph G'. Intuitively x *grants* y the ability to do α to z. In our representation



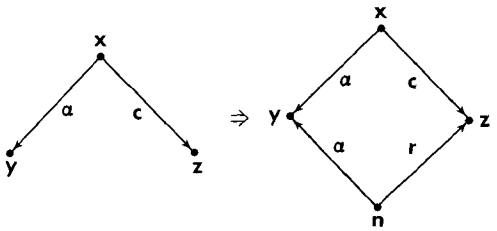
3. *Create*: Let x be any vertex in a protection graph; create allows one to add a *new* vertex n and an arc from x to n with label $\{r, w, c\}$, yielding a new graph G'. Intuitively x creates a *new* user that it can *read*, *write*, and *call*. In our representation



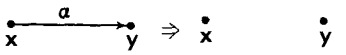
4. *Call*: Let x, y, and z be distinct vertices in a protection graph G, and let $\alpha \subseteq \{r, w, c\}$

[1] Here and in later diagrams we abuse notation by writing an explicit right as arc label (x —r→ y) to mean the arc label contains that right (i e x —γ→ y such that $r \in \gamma$)

be the label on an arc from x to y and γ the label on an arc from x to z such that $c \in \gamma$. The call rule allows one to add a *new* vertex n, an arc from n to y with label α, and an arc from n to z with label r, yielding a new graph G'. Intuitively x is *calling* a program z and passing parameters y. The "process" is created to effect the call: n can *read* the program z and can α the parameters. In our representation



5. *Remove*: Let x and y be distinct vertices in a protection graph G with an arc from x to y with label α. The remove rule allows one to remove the arc from x to y, yielding a new graph G'. Intuitively x *removes* its rights to y In our representation

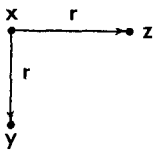


The remove rule is defined mainly for completeness, since protection systems tend to have such a rule. Moreover we expect to study properties of protection systems other than protection violations which will use remove in a crucial way. But, for the present, remove may be ignored.

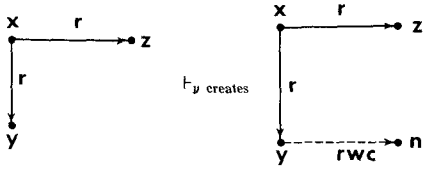
The operation of applying one of the rules to a protection graph G yielding a new protection graph G' is written $G \vdash G'$. As usual $G \vdash^* G'$ denotes the reflexive, transitive closure of $\vdash$.

An important technical point is that this system is *monotone* in the sense that if a rule can be applied, then adding arcs cannot change this. The monotone property is crucial later.

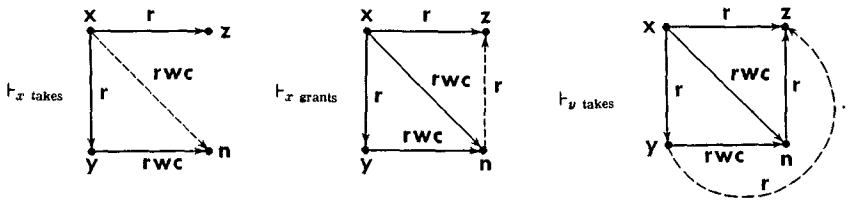
Now that we have seen the rules, let us look at their behavior. We start with a simple question: In the graph



is it possible for y to *read* z? The answer is obviously no since there is no *read* arc from y to z. But we are really asking: *Is there a sequence of rule applications that leads to a graph with a read arc from y to z*? More generally, say p *can* α q if there is a sequence of rule applications that leads to a graph with an α arc from p to q. Then to state our question more precisely, we ask: Is it true that y can *read* z? Clearly without create the answer is no since none of the operations take, grant, or call can apply. The following sequence of applications of the rules[2] shows that by using create the answer is yes:



[2] In the diagrams dashed lines are used only as a visual aid to set off the added arcs of the current operation



This example demonstrates the kind of graph-theoretic problem we are studying Our main theorem is stated in Section 2 This theorem presents a complete answer to the question: Is it true that p can α q? Indeed this theorem leads easily to a linear time algorithm for answering the question.

A final word about how this theorem contributes to our understanding of protection is as follows. Each user of a protection system needs to know:

What information of mine can be accessed by others?
What information of others can be accessed by me?

The question is vague in general, but here it is rendered in the simple question: Is it true that p can α q?

The types of protection models studied here have received considerable attention recently. Our approach is related closely to the interesting work of Harrison, Ruzzo, and Ullman [3]. They show that what can be called the "uniform safety problem" is undecidable. Interpreted as a graph model, their result says that given an arbitrary set of rules (similar in spirit to take, grant, etc.) and an initial graph, it is undecidable whether or not there will ever be an arc from p to q with label α This is a *uniform* problem in the sense that the rules are arbitrary. Even when the rules have to satisfy certain additional constraints, the results of [3] and the results of Lipton and Snyder [6] show that protection is impractically complex.

Our view here is that since the uniform protection problem is so difficult and since operating systems usually require only *one* fixed set of protection rules, the nonuniform problem should be studied. As stated before, we chose the particular take and grant system by studying the protection literature.

2 *Basic Results*

Our objective is to show that there are two simple conditions that are necessary and sufficient to determine if vertex p can α vertex q. Let G be a protection graph and $\alpha \in \{r, w, c\}$. Call p and q *connected* if there exists a path between p and q independent of the directionality or labels of the arcs. Define the predicates:

Condition 1: p and q are connected in G.

Condition 2: There exists a vertex x in G and an arc from x to q with label β such that $\alpha = r$ implies $\{r, c\} \cap \beta \neq \emptyset$, or $\alpha = w$ implies $w \in \beta$, or $\alpha = c$ implies $c \in \beta$.

Informally these conditions state that p can α q if and only if there is an undirected path between p and q (condition 1) and some vertex x α's q (condition 2).

The first step is to demonstrate the necessity of conditions 1 and 2.

LEMMA 1. *If G is a protection graph with vertices p and q and α is a label, then p can α q implies condition* 1

PROOF. Suppose p can α q and assume that condition 1 is not satisfied in $G_0, \ldots, G_i$. Then it is not satisfied in G_{i+1} since no rule application connects existing vertices not already connected Hence p and q are not connected in G_n, contrary to the assumption that p can α q. Therefore condition 1 must be satisfied. □

LEMMA 2. *If G is a protection graph with vertices p and q and α is a label, then p can α q implies condition* 2.

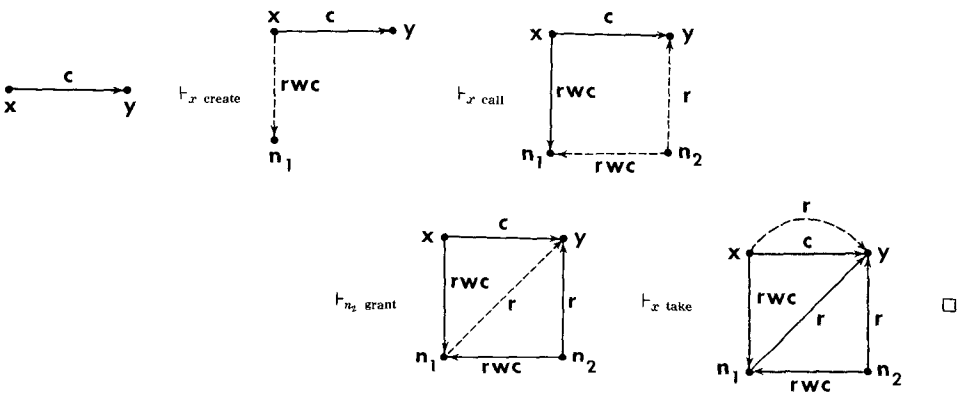
PROOF. If p can α q, either there is an arc labeled α incoming to q in G, in which case condition 2 is satisfied, or else there is no incoming arc with label α in $G_0, \ldots, G_i$ and

G_{i+1} has such a labeled arc Since take and grant merely copy arcs, $G_i \vdash G_{i+1}$ did not occur by means of take or grant. Create could not have applied; so $G_i \vdash G_{i+1}$ by application of call But no incoming α can be created that didn't previously exist as an incoming arc satisfying the lemma. Thus the arc couldn't be added and p can α q must be false, contradicting our original assumption Hence p can α q implies condition 2. □

To simplify matters later and to clear up an apparent anomaly in condition 2, we next show that if a user is allowed to *call* another user, then he is allowed to *read* him as well. It is this fact that allows us to write $\{r, c\} \cap \beta \neq \varnothing$ in condition 2 rather than just $r \in \beta$.

LEMMA 3. *In a protection graph* G, x —c→ y *implies* x —rc→ y.

PROOF. Apply the following rules:



□

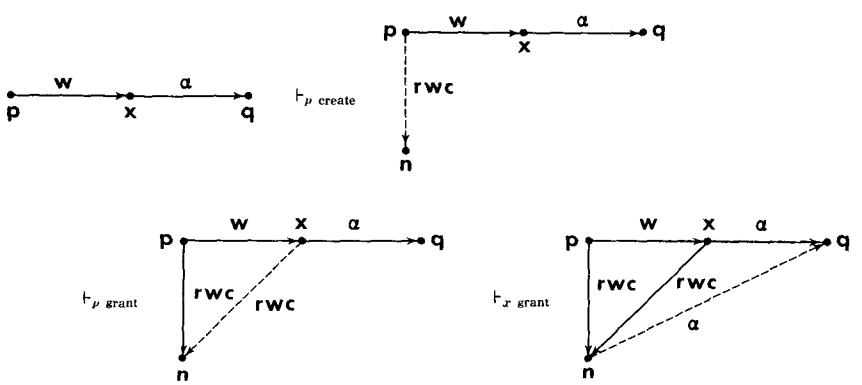
We next prove a key lemma that shows that the directionality and labels along a connected path are unimportant. Call vertices p and q of a protection graph *directly connected* if there is an arc between them independent of the directionality.

LEMMA 4 *Let p, q, and x be distinct vertices in a protection graph, let there be an arc from x to q with label α, and let p and x be directly connected. Then p can α q.*

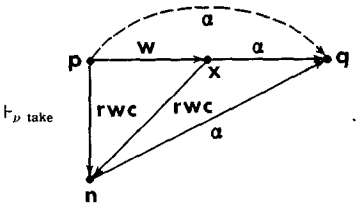
PROOF. By monotonicity, there are only six distinct cases.

Case 1.

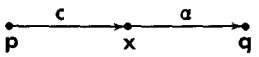
Case 2.



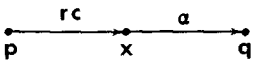
(*Case 2 continued*)



Case 3

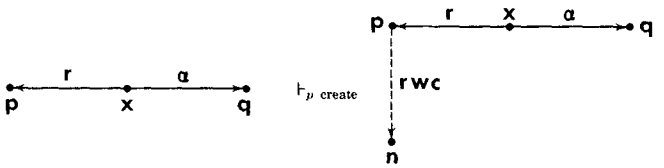


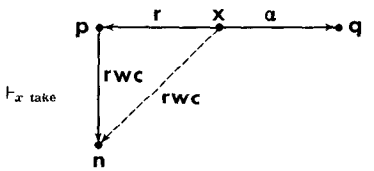
By Lemma 3 this can be written as

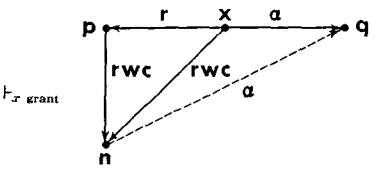


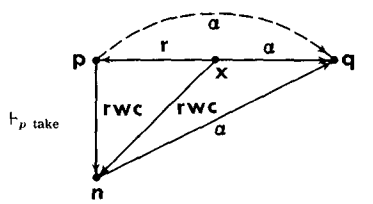
and we can appeal to case 1.

Case 4.

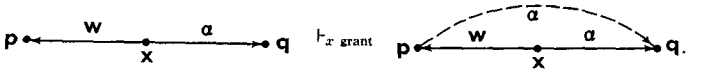




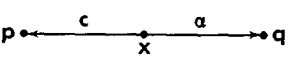




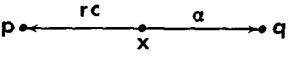
Case 5.



Case 6.



By Lemma 3 this can be written as

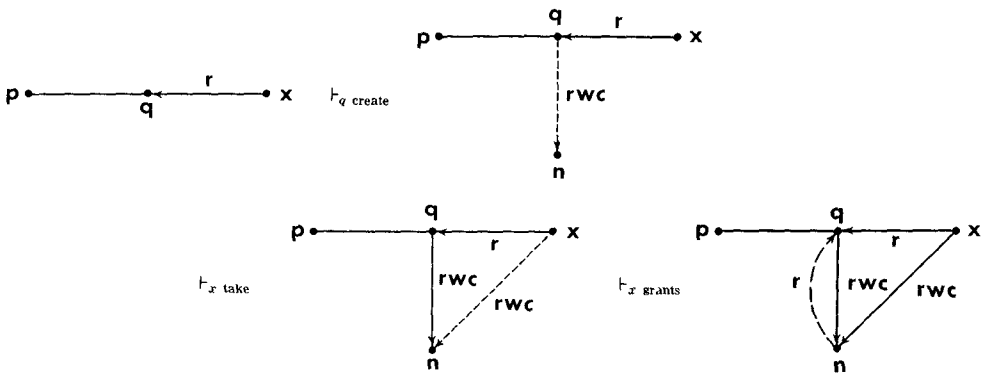


and we can apply case 4. □

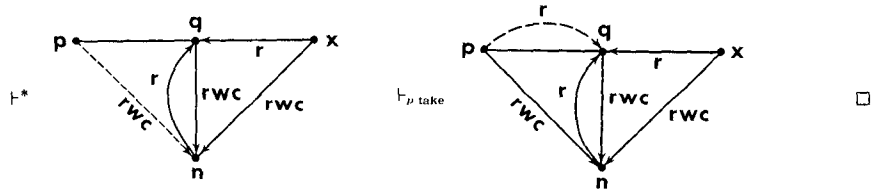
We now use Lemma 4 to prove three additional lemmas to be used in the basis of our later induction.

LEMMA 5. *Let p, q, and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $\{r, c\} \cap \gamma \neq \emptyset$. Then p can* read *q.*

PROOF. By Lemma 3, we can assume that $\gamma = r$. Then we apply the following rules:



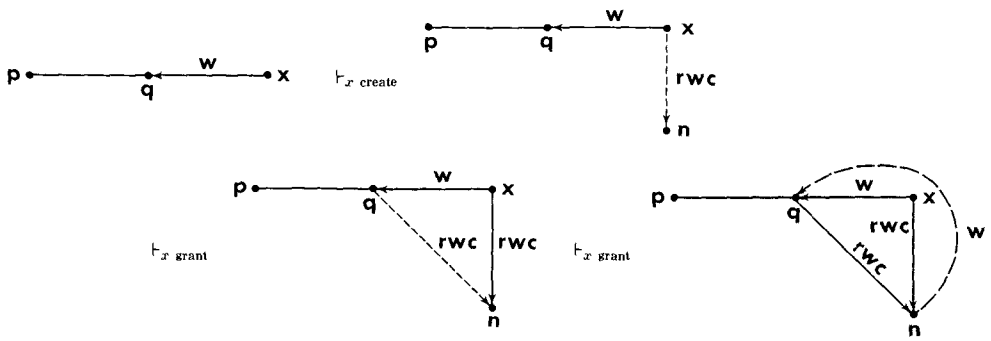
By application of Lemma 3 (on the path p, q, n), we can realize



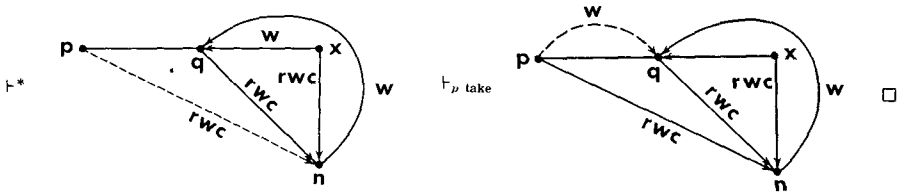
□

LEMMA 6. *Let p, q, and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $w \in \gamma$. Then p can* write *q.*

PROOF. We apply the following rules:



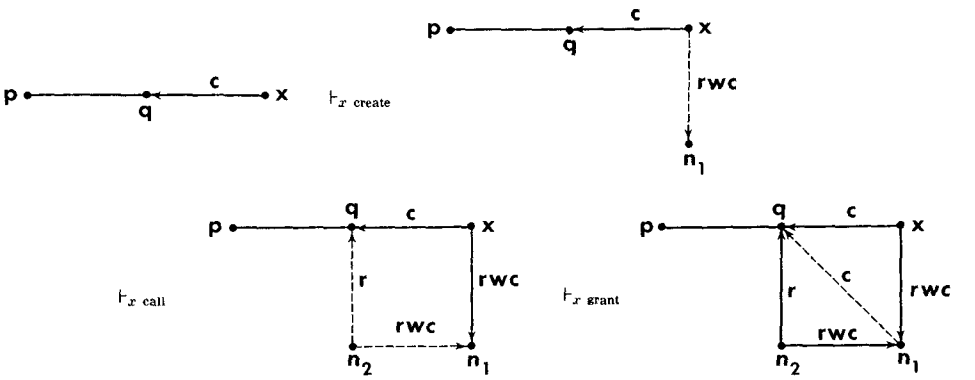
By application of Lemma 3 (on the path p, q),



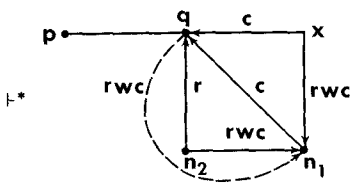
□

LEMMA 7. *Let p, q, and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $c \in y$. Then p can* call *q*

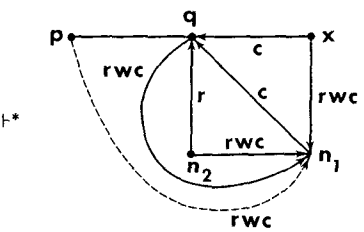
PROOF Apply the following rules:



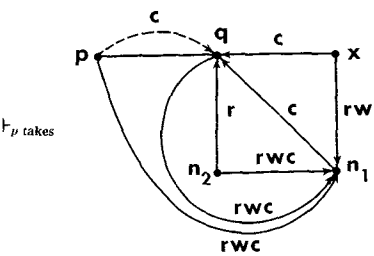
By Lemma 4 (along path q, x, n_1), we can realize



By a second application of Lemma 4 (along path p, q, n_1), we get



then



□

THEOREM. *Let p and q be distinct vertices in a protection graph and α a label. Conditions* 1 *and* 2 *are necessary and sufficient to imply p can α q.*

PROOF. Lemmas 1 and 2 demonstrate necessity; so we proceed by induction to show sufficiency. Let $p = x_n, x_{n-1}, \ldots, x_1, x_0 = q$ be the vertices on a connected path.

Basis. For $n = 1$, there are two possibilities. The x guaranteed by condition 2 either coincides with $x_1 = p$, in which case the sufficiency is immediately true, or else x and x_1 are distinct. By Lemmas 5, 6, and 7, p can α q.

Induction. Suppose the theorem is true for $n \geq 1$ and $p = x_{n+1}$ and x_{n+1} is directly connected to x_n. By hypothesis x_n can α q, and by Lemma 4 this implies x_{n+1} can αq. □

COROLLARY 1. *There is an algorithm for deciding if p can α q that operates in linear time in the size of the protection graph.*

PROOF. To verify condition 1 apply any standard connectivity algorithm. Verifying condition 2 requires no more time than scanning the in arcs to vertex q.

An obvious consequence of the constructions of this section is that it is simple to acquire the right to a given object if it can be acquired

COROLLARY 2. *If p can α q, then there exists a sequence of takes, grants, and creates containing m terms that places an arc from p to q with label α. Moreover m is linear in the length of any path between p and q*

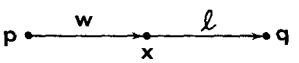
3 *Discussion*

The consequence of our main theorem is that we can precisely state the protection policy for this take-grant system.

Policy: If p can read (write) (call) q, then any user in the connected component containing p and q can attain the right to read (write) (read and call) q.

This policy may appear to be more undiscriminating than one might have expected. A primary reason for this is that our take-grant system treats all elements of the system the same whereas most protection models [3] recognize two different entities: subjects and objects. If we dichotomize the vertices of our model into subject and object sets and require (as is usually the case) that only subjects can initiate the application of our rules,[3] then the system becomes much more difficult to analyze. Such an analysis has recently been completed and appears in Jones, Lipton, and Snyder [5]. It should be noted that in the dichotomized model there are protection graphs that satisfy conditions 1 and 2 for which p can α q is false.

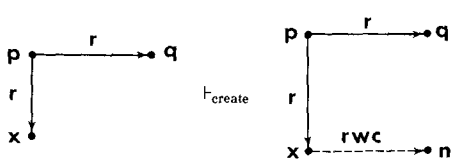
In addition to completing the subject/object analysis, there are other problems to be studied. For example, consider a protection graph G where there is an arc from vertex x to vertex q with label l where l is a new type of label. We wish to know if p can l q, i.e. if there is a series of takes, grants, creates, and calls that leads to a graph with an arc from p to q with label l. The key to this problem is that while label l can be taken and granted it has no special role(s) as r, w, and c do. The label l is simply something that is passed around, and that is all. A graph such as



shows that our theorem is no longer true under these new assumptions.

Another way to modify our system is to control the amount of cooperation necessary to obtain a particular right. With each rule application, the vertex that is denoted x in our definitions will be called a *conspirator*.

Thus in



x is a conspirator. Then an interesting question is: If p *can* α q, can it do so with at most m conspirators? One might then hope to attach some kind of likelihoods in a precise way to whether or not a system is secure.

In general there are many other problems to be studied. All of these problems are in a

[3] The restriction that only subjects can initiate protection rules is enforced by requiring the x vertex in our rule definitions to be a subject while all other vertices may be either subjects or objects.

sense generalizations of transitive closure. The key and most important aspect of this generalization is that the most interesting rules allow "growth," i.e. the addition of new vertices. It appears that understanding the structure of such problems is interesting beyond its application to the study of protection models.

ACKNOWLEDGMENT. We gratefully acknowledge the help of our colleague Anita K. Jones for her role in developing this model, and the careful comments of a referee. We also acknowledge Mike A Harrison for several important comments.

REFERENCES

1 COHEN, E Problems, mechanisms and solutions Ph D Th (in progress), Carnegie-Mellon U , Pittsburgh, Pa , 1976
2 DENNING, P J , AND GRAHAM, G S Protection—principles and practice Proc AFIPS 1972 SJCC, Vol 40, AFIPS Press, Montvale, N J , pp 417–429
3 HARRISON, M A , RUZZO, W L , AND ULLMAN, J D On protection in operating systems Operating Syst Rev (ACM SIGOPS Newsletter), *9*, 5 (1975), 14–24
4 JONES, A K Protection in programmed systems Ph D Th , Carnegie-Mellon U., Pittsburgh, Pa , 1973.
5 JONES, A K , LIPTON, R J , AND SNYDER, L A linear time algorithm for deciding subject-object security Proc 17th Annual FOCS Conf , Houston, 1976, pp 33–41.
6 LIPTON, R J , AND SNYDER, L Synchronization and security In preparation, 1976

RECEIVED JUNE 1976, REVISED OCTOBER 1976