# I. Inert Rights and Conspirators in the TAKE/GRANT System

# II. Safety in Protection Systems

Timothy A. Budd
Richard J. Lipton

Research Report #126

Supported in part by NSF Grant MCS76-81486 and NSF Grant MCS77-12517.

## 1. INTRODUCTION

Interest in the modeling and formal analysis of operating system protection mechanisms has increased in the last few years [2-7,11-14]. In [7] it was shown that for arbitrary systems the sort of questions we are interested in asking, such as whether rights can be passed to unauthorized persons, are generally undecidable. On the other hand, in [12] it was shown that for a system which had previously been proposed in the literature [4,11], such questions could be decided in linear time.

This paper will be divided into two parts. In the first part, we will demonstrate new results concerning the latter system mentioned above, answering some questions proposed in [12]. In the second part, we attempt to bridge the gap between this specific system and general "undecidable" systems. We define a relatively large class of protection systems based on the model of [12], and using some well known results from language and automata theory we are able to prove that for many of these systems the safety question can be answered in linear time.

## 2. RESULTS CONCERNING THE SUBJECT/OBJECT PROTECTION SYSTEM

The Subject/Object protection system has been analyzed previously in the literature [4,11,12]. We will briefly outline the details and pertinent results which are known about that system here.

The state of the system is represented by a graph G where the vertices of the graph are of two kinds: subjects (intuitively processes) and objects (intuitively files). Directed arcs between vertices then represent rights which one vertex has over another. Arcs are labeled with letters from some finite alphabet Σ. For instance, an arc labeled r represents the ability to *read* the indicated vertex, and one labeled w the ability to *write*. We will refer to arcs labeled with other elements of Σ as being *inert*, since they can be passed around but play no special role in the rewriting rules.

We model dynamic change in the system by a set of *rewriting rules*, which by means of local changes transform a graph G into a new graph G'. There are three rewriting rules, shown in figure 1. Solid dots represent subjects, open ones objects, and crossed dots represent either subjects or objects. α indicates any element of Σ.

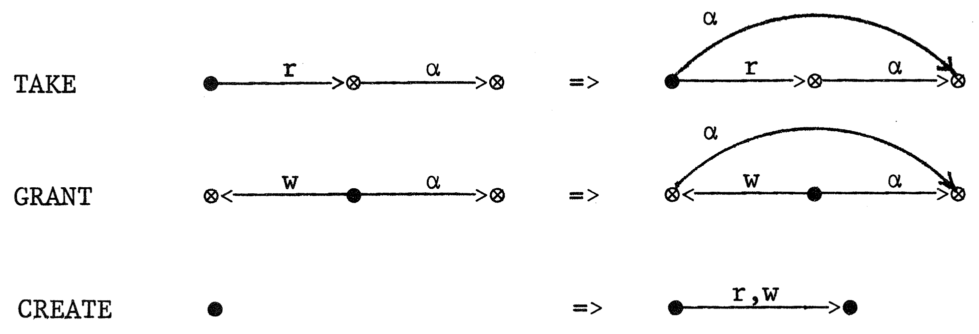


*Figure 1*

We shall use the phrase P can $\alpha$ Q to mean that by a finite sequence of rewritings we can construct an arc between P and Q labeled $\alpha$.

If two subjects are connected by an arc labeled r or w, we shall say they are directly connected. We define a *block* in a graph G to be any maximal directly connected subgraph.

For each simple path in G we can associate a word over the alphabet $\overrightarrow{\Sigma} \cup \overleftarrow{\Sigma}$ in the obvious way, for instance the path shown in figure 2 has the word $\overrightarrow{r}\,\overleftarrow{r}\,\overrightarrow{w}\,\overrightarrow{r}\,\overleftarrow{w}$

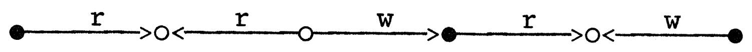


*Figure 2*

Let E be the union of the following regular languages $\{\overrightarrow{r}^*, \overleftarrow{r}^*, \overrightarrow{r}^*\overrightarrow{w}\overleftarrow{r}^*, \overrightarrow{r}^*\overleftarrow{w}\overleftarrow{r}^*\}$. We say there is a *bridge* between two blocks A and B if there is path with associated word in E between some subject in A and some subject in B.

We can now state the results proved in [12]. Let $\alpha \in \{r,w\}$.

*Theorem 1:* [12] If P, Q and X are in the same block and X is connected to Q by an arc labeled $\alpha$, then P can $\alpha$ Q.

*Theorem 2:* [12] If P, Q are subjects with some subject having an edge to Q with label $\alpha$, then P can $\alpha$ Q *iff*

*Condition 3:* There exists a sequence of blocks $B_1, B_2, \ldots, B_k$ with P in $B_1$, Q in $B_k$, and for each $i = 1,\ldots,k-1$ there is a bridge from $B_i$ to $B_{i+1}$.

In the course of proving these two theorems, there are two lemmas which are instrumental. We shall have occasion to refer to each of them later, hence we restate them both here.

*Lemma 1:* (Proved as Lemma 3 in [12].) Let P, Q and X be distinct vertices in a protection graph. Assume $\alpha \epsilon \Sigma$. Let there be an arc from X to Q with label $\alpha$ and let P and X be directly connected. Then P can $\alpha$ Q.

*Lemma 2:* (Proved as Lemma 7 in [12].) If P and Q are subjects connected by a path with word in E, then there is a sequence of *takes*, *grants* and *creates* such that P and Q can become directly connected.

Two open problems proposed in [12] concerned extensions of this system to incorporate *inert rights* and *conspirators*.

*Inert Rights*

As mentioned previously, we can characterize an inert right as simply a right which plays no special role in the rewriting rules, as do r and w. Analogous to the first two theorems, we have two theorems concerning inert rights.

*Theorem 3:* If there exists some vertex X with inert rights $\eta$ to some vertex Q, and P and X are in the same block, then P can $\eta$ Q.

*Proof:* Since P and X are in the same block, there must be a path between them which is composed entirely of subjects. A simple induction on the length of this path using Lemma 1 then serves to provide the result.

*Theorem 4:* Given a subject P and a vertex X (either subject or object) then for all labels $\alpha$ in $\Sigma$, P can $\alpha$ X *iff* there exists some subject Q connected by a path with word $\vec{r}^{*}\vec{\alpha}$ with X, and Condition 3 is true for P and Q.

*Proof:* Assume Condition 3 is true for P and Q. Lemma 2 tells us that P and Q can become directly connected. By a sequence of *takes*, Q can $\alpha$ X. Then by Lemma 1 P can $\alpha$ X.

To prove the other direction we assume P can $\alpha$ X. We create a new system by modifying the protection graph G as follows.

The right which P obtained must have been passed by a sequence of *takes* and *grants* from some vertex Y which had the ability to $\alpha$ X in the original graph. We remove this arc, and replace it by a *read* arc to a new vertex X'; furthermore, we give X' the ability to read a new vertex X" (see figure 3).

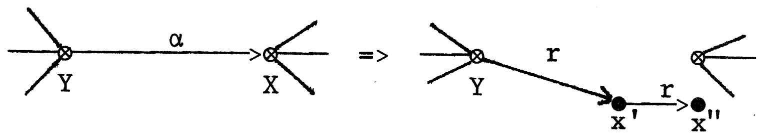


*Figure 3*

By following the same sequences of moves as before, P can r X' in the new system. A single take shows P can r X". Theorem 2 then asserts that Condition 3 must be true for P and X". We then have two cases.

<u>*Case 1.*</u> Y is a subject.

In this case, since Y and X" are in the same block, Condition 3 must be true also of P and Y. Let Q = Y and we are done.

<u>*Case 2.*</u> Y is an object.

In this case, X' and X" are in a block to themselves. There must be a block between P and X' such that the block and X' are connected by a path in E; furthermore, this path ends in $\vec{r}$. From this we can deduce the path must have been $\vec{r}^{\,*}$. Let Q be the subject in that block connected to this path, Theorem 2 asserts that Condition 3 must be true for P and Q.

Since Q is connected to X in the original graph by a path with word $\vec{r}^{*}\vec{\alpha}$ we are finished.

We note that this theorem extends Theorem 2 in two different directions, by telling us under what conditions P can obtain rights to objects as well as subjects, and by telling us under what conditions P can obtain inert rights.

As a corollary to this, we see that given a vertex Q and an arc labeled α into Q, there is an algorithm which works in linear time in the size of the protection graph which enumerates all the vertices which can obtain α rights to Q.

## *Conspirators*

It should be obvious that in all but the most trivial cases it is not possible for a vertex P to acquire rights to another vertex Q, even if all other conditions are met, without the assistance of other intermediary vertices. If by a finite sequence of rewritings P can acquire the ability to α Q, then we define a *conspirator* to be any vertex which was required to perform a *take*, *grant* or *create* in that sequence. Notice this does not automatically include all vertices on the path between P and Q; for instance, in figure 4 P can w Q without the assistance of any conspirators.

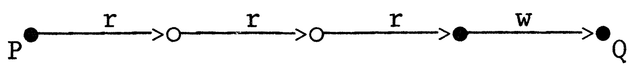


*Figure 4*

We define a vertex weighing function by assigning each vertex a value from the table given in figure 5. Where more than one value would apply, we choose the lowest. Objects are assigned weights the same as subjects, except that where we assign a subject the weight of one we assign an object a weight of infinity. Vertices with indegree or outdegree one are assigned a weight of zero.

Notice that given a path this function assigns a weight to each vertex depending upon the label on the entering and leaving arcs, the type of the vertex (subject or object) and the label on any other arcs which may be connected to the vertex. For this given path we define the *path weight* to be the sum of these values.

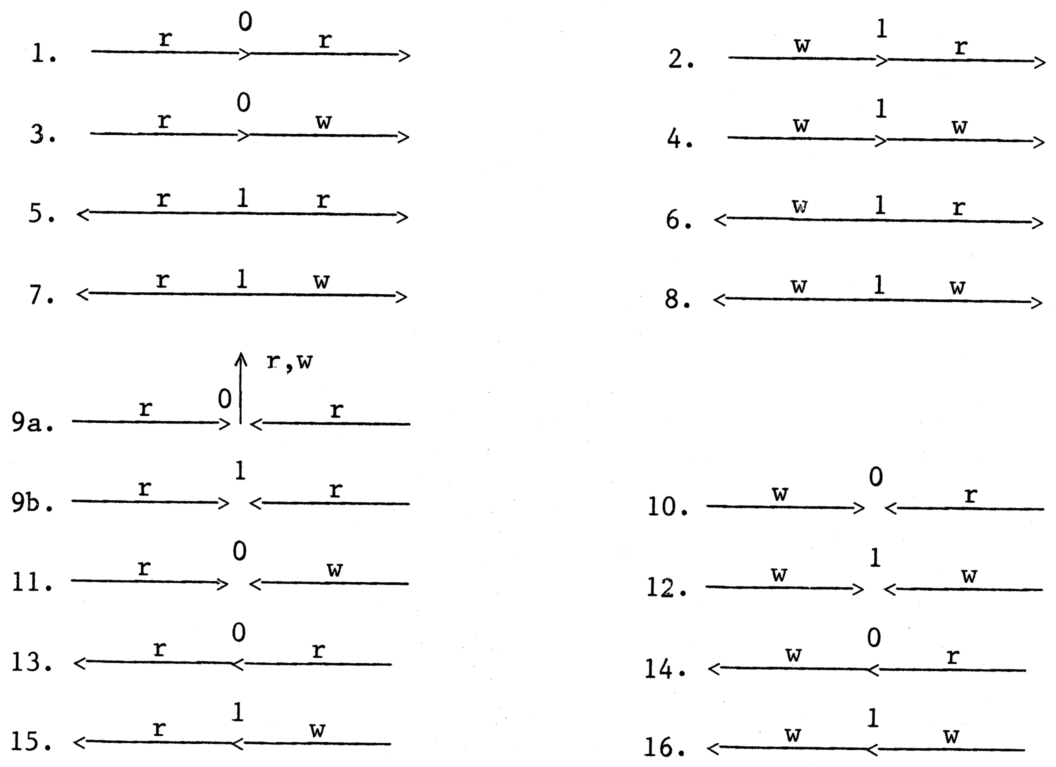


*Figure 5*

*Theorem 5:* Given a subject P and a vertex Q, if P can acquire the $\alpha$ rights to Q, then it must require at least M conspirators, where M is the minimum path weight of all paths between P and Q.

*Proof:* The proof will be in three steps.

*Step 1.* We are given a graph G which consists solely of a vertex X which can $\alpha$ a vertex Q, and a cycle free path between a subject P and X which does not include Q. (See figure 6.)

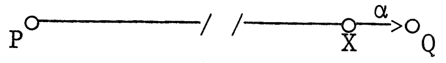


*Figure 6*

We then show that P can $\alpha$ X only with the help of at least M conspirators where M is the path weight.

First we note that if any vertex was marked with a value of infinity, then by Theorem 4 P cannot $\alpha$ Q since we have a sequence of objects but no bridge over them. If P can $\alpha$ Q we will show that all the M vertices marked with a one must be conspirators. First we need the following lemma.

*Lemma 3:* Given a graph as above, if P can $\alpha$ Q then if we replace all the vertices marked zero with object vertices, P can still $\alpha$ Q.

*Proof:* Recalling Theorem 2, we need only show that we have not destroyed any bridges and that any new sequence of object vertices we have created are also bridges.

I shall show the proof for the case where we had a bridge in $\{(\vec{r})^*, (\vec{r})^*\overleftarrow{w}, (\vec{r})^*\overleftarrow{w}(\overleftarrow{r})^+\}$, and in which we add object vertices on the left. The remaining cases can be shown in a similar fashion.

Assume we had a bridge with word $(\vec{r})^*$. The applicable cases are 1, 3, 9a and 11, and all give us words still in E; hence we still have a bridge.

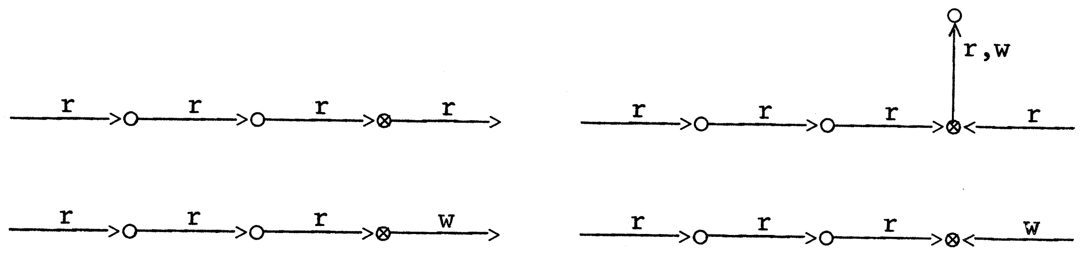


*Figure 7*

Next assume the bridge had word $(\vec{r})^*\overleftarrow{w}$. The only applicable case is 14, which stills gives us a word in E.

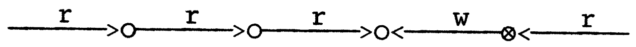


*Figure 8*

Lastly, assume our bridge had word $(\vec{r})^*\overleftarrow{w}(\overleftarrow{r})^+$. The only rule that would apply is 13, which again still leaves us a bridge.

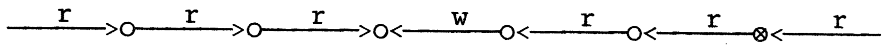


*Figure 9*

The other cases can be shown in a similar fashion. I claim now that since the original bridge could have been null, we have also shown that we have not created any sequence of objects which is not a bridge.

Hence by Theorem 4, P can still $\alpha$ Q.

By the above lemma, if we replace all the vertices marked with a zero with objects, P can still α Q. Next, we simply observe that if we make any subject vertex marked with a one into an object, we will have created a path with a word not in E, hence not a bridge and hence, by Theorem 4, P cannot α Q.

In a certain sense, M then represents the "minimum" number of conspirators required for P to obtain the α rights to Q.

*Step 2*. Given a graph G which consists solely of the following: a subject vertex P connected by a cycle free path to a vertex Q, with another vertex X having α rights to Q, then for P to obtain α rights to Q requires at least M conspirators, where M is the path weight on the path between P and Q.

Assume P can obtain the rights using less conspirators, that is, the graph shown in figure 10 is somehow "easier" (in terms of the number of conspirators) than the one shown in figure 11, which we know requires at least M conspirators.

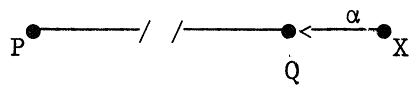


*Figure 10*

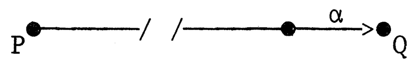


*Figure 11*

Since *takes* and *grants* only move the tail of the arc, we can convert the first graph into that shown in figure 12

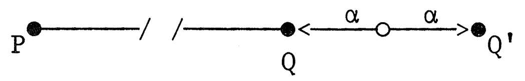


*Figure 12*

and by following the same sequences of moves as before, P can obtain the $\alpha$ rights to Q' in less than M conspirators. But the path weights are monotonic in the length of the path, hence the path weight of this graph cannot be less than that of figure 11, which we know to take M conspirators. This contradiction gives us our result.

*Step 3.* The main theorem.

*Proof:* Assume P can $\alpha$ Q using only n conspirators with $n < m$. P can only $\alpha$ Q in virtue of having taken the right or having been passed it from some vertex X. In fact, in the final graph there must be a path $D = X_0, X_1, \ldots, X_k, P$ such that $X_0$ could $\alpha$ Q in the original graph and $X_i$ can $\alpha$ Q in virtue of having been passed the right or taken it from $X_{i-1}$. Since the rewriting rules preserve connectedness and do not connect disjoint components there must have been some path between P and Q in the original graph. The form of D depends only upon this original path. But by hypothesis the original path required more than M conspirators to move the right over it. This contradiction gives us the main theorem.

## Cost Metrics

The function used in the previous section is an example of a much more general class of *path-vertex weighting functions*. We characterize a Path-Vertex Weighting Function as any non-negatively valued function which assigns a vertex a value depending upon 1) the label on the arc entering the vertex, 2) the label on the arc leaving the vertex, 3) the nature of the connections to the immediate neighbors of the vertex and 4) any a priori assigned value associated with the vertex (type, degree of difficulty in passing rights over this vertex, measure of reliability, etc.).

The reader is assumed to be familiar with Dijkstra's single source shortest path algorithm [1]. The complexity of this algorithm is $O(V^2)$; however, Johnson [10] has shown how, using a treesort-like approach, one may improve Dijkstra's algorithm to $O(E)$. We need not concern ourselves here with the details of either algorithm other than to note that they both operate by taking the known shortest distance to a vertex X (see figure 13), adding on the distance from the vertex X to the vertex Y to obtain the distance to the vertex Y.

Hence, all we need to do is redefine the term "distance from X to Y" to be the vertex-weight of X, and we have enough features of a metric (i.e., Monotonicity) to allow us to use this procedure.

*Theorem 6:* Given an arbitrary cost metric and an initial vertex P, there is an algorithm which works in linear time in the size of the protection graph to determine for each vertex in the graph the cost for P to obtain rights to that vertex.

Notice this algorithm not only allows us to answer the conspiracy question as posed in [12], but allows for very broad generalizations of that problem. For instance, suppose we can somehow establish *a priori* likelihoods that any individual will contribute to a conspiracy. As an example, if I covertly wish to obtain rights to a file it might be easier to go through five unreliable processes than to use one certified reliable process.

Now, instead of just giving each potential conspirator a weight of one, as we did previously, let us give each conspirator a weight proportional to this likelihood; say, a number from one to one hundred.

We can then answer such questions as given a vertex P, what files can he obtain rights to at a cost of less than 1000 units. Such ideas can be used to provide more of a statistical measure on the security of a system.

## 3. EXTENSIONS TO MORE GENERAL PROTECTION SYSTEMS

In view of the undecidability results already mentioned, it is interesting that the system studied in the last section has sucn easy decision properties. In this section, we attempt to bridge the gap between this particular system and completely general systems by demonstrating other systems which also have simple decision procedures. In doing so, we uncover some relationships between protection systems and language theory.

### *Safety in Protection Systems*

With regards to the work of Harrison, Ruzzo and Ullman [7], we can define two general questions to be analyzed with respect to protection systems.

*Question 1:* The *safety* question.

Given a protection system G and two objects $X_i$ and $X_j$ in that system, if we introduce the right of $X_i$ to $\alpha$ $X_j$, what other objects can thereby obtain the rights to $\alpha$ $X_j$.

*Question 2:* The *extended safety* question.

Given a protection system G and two objects $X_i$ and $X_j$ in that system, if we introduce the right of $X_i$ to $\alpha$ $X_j$, what potential changes will this produce in the entire system.

Harrison, Ruzzo and Ullman have shown [7] that for general protection systems these problems are undecidable. For certain restricted types of systems they were able to give decision procedures for the safety problem;

however, their procedures worked in exponential time.

The remainder of this paper will be devoted to the classifications of differing protection systems, indicating some classes for which polynomial or linear time results can be shown for the above mentioned questions.

Our paradigm of a protection system will be as follows:

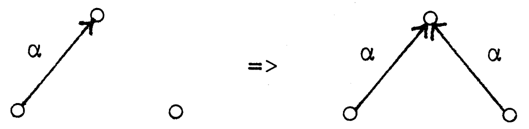
We are given N objects in the system $(X_1,\ldots,X_n)$ where each $X_i$ is of type $T_i$ which is an element of some finite alphabet T. Between any two objects there may be an arbitrary number (possibly empty) of *rights*, where each right is indicated by an element from some finite alphabet $\Sigma$.

At any time we indicate the current status of the system by a graph G where each object is represented by a vertex and each right by a labeled directed arc.

The differences we will emphasize in classifying protection systems will be in the rules they use for adding or deleting arcs from an existing graph. These we will call the *transition rules*.

If starting from some initial configuration by a finite number of applications of the transition rules we can connect a vertex X to a vertex Y by an arc labeled $\alpha$, we will say that in the initial configuration X can $\alpha$ Y.

We will not consider systems which have rules roughly equivalent to "If I have a right to something, I can give it to anyone I choose," or graphically,



Such systems we can refer to as "loose." For these systems, the safety question tends to be either trivial or nonsensical. For example, if I can obtain the $\alpha$ right to Z, then anybody can obtain the $\alpha$ right to Z.

The hazardous effects of having a loose protection system have generally not been recognized; for instance, all the examples given in [4,7] suffer from being loose.

Notice that there is a simple relationship between systems represented in this graphical format and systems represented in the *access matrix* format of [4,6,7].

## *Grammatical Protection Systems*

We will call a protection system *grammatical* if for each right $\alpha \in \Sigma$ there is a grammar L and start symbol S such that given two vertices X and Y, X can $\alpha$ Y *iff* X and Y are connected by a path in L(S).

We will illustrate this concept by demonstrating a class of protection systems and showing them to be grammatical; from this we can obtain a polynomial time solution to the extended safety question.

## *General Arc Passing Systems*

Working within the model described previously, we will define a General Arc Moving system to be a protection system with transition rules of the following form

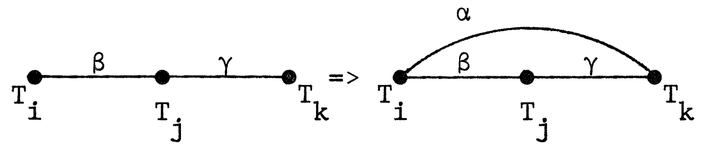


*Figure 14*

where the types of $T_i$, $T_j$ and $T_k$ indicate the necessary types for the vertices and $\alpha$, $\beta$ and $\gamma$ are rights. The directionality on the arcs must be specified, but they are here omitted for generality.

We obtain a grammar L by defining a new production for each rewriting rule. For each rule such as that in figure 14, we define a production of L as follows. If they do not already exist, we introduce three non-terminals A, B and C $\in$ T×R×T such that A corresponds to an arc labeled $\alpha$ between vertices of type $T_i$ and $T_k$, and in a similar fashion B and C are defined. We then have the production

$$A \rightarrow BC \ .$$

Note that the nonterminals A, B and C encode both the nature of the right and the type of the vertices that the right connects. For each non-terminal A, we create its terminal counterpart a and add the production $A \rightarrow a$.

We then have the following lemma:

*Lemma 4:* Given two vertices P and Q of types $T_p$ and $T_q$, respectively, P can $\alpha$ Q *iff* there exists a path between P and Q in $L((T_p,\alpha,T_q))$.

*Proof:* If P and Q are connected by a path with word in $L((T_p,\alpha,T_q))$, then the derivation of that word gives us a constructive method by which we can join P to Q by an arc labeled $\alpha$; hence, P can $\alpha$ Q.

The proof the other way will be by induction on the number of applications of the transition rules which lead to P being able to $\alpha$ Q.

If this number is zero, that is, P had the rights to Q in the original graph, then we trivially have our result. Hence, we assume P did not originally have the $\alpha$ rights to Q, and that it took n applications of the transition rules for P to obtain that right.

The very last application of a transition rule must have been for P to get the ability to $\alpha$ Q from some vertex X (see figure 15). This must have been permitted in virtue of some right $\beta$ between P and X and some right $\gamma$ between X and Q and there being a transition rule as shown in figure 14.

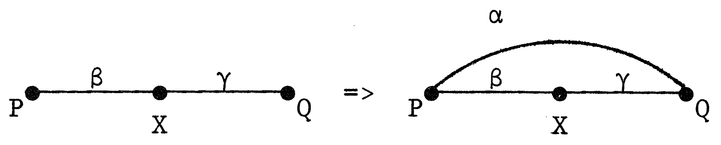


*Figure 15*

Now, it took less than n applications of the transition rules to form the arcs between P and X and between X and Q; hence, by the induction hypothesis there must have been a path between P and X in $L(T_p,\beta,T_x)$ and between X and Q in $L(T_x,\gamma,T_q)$. But associated with the transition rule shown in figure 15 is the production $(T_p,\alpha,T_q) \rightarrow (T_p,\beta,T_x)(T_x,\gamma,T_q)$. Hence, it must be the case that Q and P were connected by a path with word in $(T_p,\alpha,T_q)$.

We can note the similarity between grammars in this form and context free grammars in Chomsky Normal Form [9]. In view of this, and the relationship between parsing and protection systems demonstrated by Lemma 4, it is too much to expect the safety question for arbitrary arc passing systems to be answered in linear time. However, we can demonstrate a polynomial time result as shown by the following theorem.

*Theorem 7:* The extended safety question can be answered for a general arc moving protection system in $O(N^{2.81})$.

*Proof:* For this result we assume the protection network is kept in an N by N matrix (call it M), similar to the access matrix of [3,6,5]. We then define a matrix "multiplication" operation by substituting production reduction ($BC = A$ *iff* $A \rightarrow BC$) for scalar multiplication and set union for scalar addition in the standard matrix multiplication algorithm.

We next observe that since the lower triangular portion of M is the inverse of the upper triangular part, by suitably adding production rules we can just work with the upper triangular part of M. Hence, we have reduced the problem to that of finding the transitive closure of an upper triangular matrix with respect to our matrix multiplication operation. Valiant [15] has shown how this can be accomplished in $O(N^{2.81})$ operations.

To give a solution to the extended safety question, we simply perform this operation twice, once with and once without the additional arcs. Comparing the results then gives us our answer.

*Example 1.* A non-regular Arc Passing System.

In this example, we are just concerned with the movement of *read* privileges. Assume we have a right called the *indirect* right, such that if X has the indirect right to Y, and Y can read Z, then in effect X can read Z. Next there is the *request* right, which says that if X can *request* of Y, and Y has indirect rights to Z, then X can obtain indirect rights to Z. (Notice here, as in the take/grant system [12], we take the worst case approach by assuming requests are always granted.) Finally, if X has read rights to Y, and Y has request rights to Z, X can obtain request rights to Z.

The transition rules are shown in figure 16. If we let A represent *read*, B *indirect* and C *request*, we obtain the following grammar.

$$A \rightarrow BA$$

$$B \rightarrow CB$$

$$C \rightarrow AC$$

This obviously is an arc passing grammar; hence, Theorem 7 gives us a method for solving the safety question. Furthermore, it can be shown [8] that this grammar is not regular.

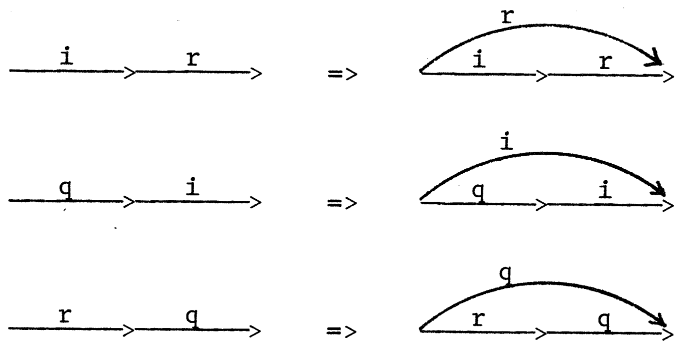


*Figure 16*

## *Regular Grammatical Systems*

If it happens that for each right the language generated by the grammar associated with a grammatical protection system is regular, we will say the system is a *regular grammatical system*.

Regular grammatical systems are important on account of the following theorem.

*Theorem 8:* For regular grammatical systems, the *safety* question can be answered in linear time in the size of the protection graph.

*Proof:* We prove this result by appealing to the fact that regular grammars can be recognized by finite state automata. Assume for a given grammar G we have an automata with T states that recognizes L(G). And assume our protection graph has B vertices. We then construct a new graph with B×T vertices, where there is an arc from $(G_i,T_j)$ to $(G_k,T_l)$ *iff* there was an arc from $G_i$ to $G_k$ in the original graph, and if we were in state $T_j$ at the point $G_i$ that arc would carry us to state $T_l$.

Starting from the vertex X and using depth-first search on the original graph, we see we can construct this new graph in O(E) operations. Again using a depth-first search on the new graph, we mark those vertices we encounter which are in designated final states for the automata. These are then the only vertices which can obtain rights to X. Again we have a complexity of O(E) operations.

We wish then to characterize protection systems which have regular languages.

A class of grammars which seem to arise quite frequently are what we call *non-discriminating* grammars. Informally, we will say a protection system is non-discriminating if all the transition rules are of the form "If X and Y are constructed by an arc with some right $\gamma$, and Y has any right to Z, then X can obtain that right to Z."

The name is meant to imply the fact that we don't discriminate between rights in the second context.

Formally, we will say a protection system is nondiscriminating if it has a nondiscriminating grammar. We define a nondiscriminating grammar as follows.

There are five types of nonterminals, $A_1,\ldots,A_{ka}$, $B_1,\ldots,B_{kb}$, $C_1,\ldots,C_{kc}$, $D_1,\ldots,D_{kd}$ and Z. We allow productions of the following forms (greek letters represent strings of terminal symbols).

Any nonterminals of type A, B, C or D can produce a finite string of terminal symbols.

$$A_i \rightarrow \alpha_{aij} \quad B_i \rightarrow \alpha_{bij} \quad C_i \rightarrow \alpha_{cij} \quad D_i \rightarrow \alpha_{dij}$$

A's, B's, C's and D's are allowed productions of the following forms:

$$\begin{array}{ll}
A_i \rightarrow \beta A_j & B_i \rightarrow B_j\gamma \\
C_i \rightarrow ZA_j & C_i \rightarrow ZB_j \\
C_i \rightarrow A_j & C_i \rightarrow B_j \\
D_i \rightarrow A_jZ & D_i \rightarrow B_jZ \\
D_i \rightarrow A_j & D_i \rightarrow B_j
\end{array}$$

For Z we allow productions of the following form:

$$\begin{array}{ll}
Z \rightarrow A_i & Z \rightarrow B_j \\
Z \rightarrow C_i & Z \rightarrow D_i
\end{array}$$

$$Z \rightarrow C_iD_j$$

$$Z \rightarrow ZZ$$

$$Z \rightarrow \Lambda$$

*Theorem 9:* Nondiscriminating grammars are regular.

*Proof:* We wish to show that starting from any nonterminal, the language produced from this grammar is a regular event. The proof for nonterminal of the first four classes quickly reduces to showing the language produced from Z is a regular event; hence, we show only this case.

Notice first that the nonterminals A (B) form by themselves a right linear (left linear) language and hence associated with every nonterminal $A_i$ or $B_i$ we have a regular event which represents the language that can be generated from that start symbol.

Let us consider first the set of sentential froms that can be generated from Z using only productions of the type $Z \rightarrow A_i$, $A \rightarrow B_i$, $Z \rightarrow C_i$ and $Z \rightarrow D_i$.

Let us construct a regular event L as follows. If there are productions $Z \rightarrow D_i$, $D_i \rightarrow A_jZ$ or $D_i \rightarrow B_jZ$ and $A_j \Rightarrow \omega$, $B_j \Rightarrow \nu$ then both $\omega$ and $\nu$ are in L, and nothing else is in L. We can do a similar trick with $C_i$ to form a regular event R; finally, we can define a regular event F as follows: If there are productions $Z \rightarrow A_i$ and $A_i \Rightarrow \omega$, then $\omega$ is in F; similarly for $B_i$, $C_i$ and $D_i$.

It should be obvious that the set of sentential forms Z can generate is $L^*(Z!F)R^*$. That is, everything in this form can be generated from Z (using only the productions we have indicated) and nothing else can.

Now assume we have a production $Z \rightarrow ZZ$ (the proof in the case where we don't have this production is easier and won't be given here). Consider what can happen with one application of this rule.

$$\begin{aligned} Z &\Rightarrow L^*ZR^* \\ &\Rightarrow L^*ZZR^* \\ &\Rightarrow L^*(L^*(Z!F)R^*)(L^*(Z!F)R^*)R^* \\ &\Rightarrow (L^*(Z!F)R^*)(L^*(Z!F)R^*) \end{aligned}$$

A simple induction argument can be used to show that the set of sentential forms Z can generate is then $(L^*(Z!F)R^*)^*$.

We now wish to add the productions $Z \rightarrow C_iB_j$.

Let us look at what we can generate with one application of this rule.

$$\begin{aligned} Z &\Rightarrow (L^*(Z!F)R^*)^*L^*ZR^*(L^*(Z!F)R^*)^* \\ &\Rightarrow (L^*(Z!F)R^*)^*L^*C_iD_jR^*(L^*(Z!F)R^*)^* \\ &\Rightarrow (L^*(Z!F)R^*)^*L^*ZN_1N_2ZR^*(L^*(Z!F)R^*)^* \\ &\Rightarrow (L^*(Z!F)R^*)^*(L^*(Z!F)R^*)^*N_1N_2(L^*(Z!F)R^*)^*(L^*(Z!F)R^*)^* \\ &\Rightarrow (L^*(Z!F)R^*)^*N_1N_2(L^*(Z!F)R^*)^* \end{aligned}$$

Associated with each nonterminal pair $N_1N_2$ is a regular event. Let us call the union of all such regular events W. Hence, we have that the set of sentential forms Z can generate with one application of a production $Z \rightarrow C_iD_j$ is simply

$$(L^*(Z!F)R^*)^*W(L^*(Z!F)R^*)^* .$$

Let $A^r$ be A repeated r times, with $A^0 = \Lambda$. We now want to show that the set of sentential forms generated by Z using n applications of productions of the form $Z \rightarrow C_iB_j$ is

$$((L^*(Z!F)R^*)^*W)^n(L^*(Z!F)R^*)^*$$

which is equal to

$$= (L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^n$$

The proof is by induction. We have just shown it true for 1, hence we assume it is true for n and show it is true for n+1.

First we note that if P,Q <= n then the set of sentential forms we can derive from

$$(L^*ZR^*)(L^*ZR^*)$$

where the left Z is expanded using p applications of the rule and question and the right Z using q, is just

$$L^*(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^p R^* L^*(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^q R^*$$
$$= ((L^*(Z!F)R^*)^*W)^p(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^q$$
$$= (L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^{p+q} .$$

From this we see that the set of sentential forms that can be generated using n applications of the productions in question starting from $(L^*(Z!F)R^*)^*$ is just $(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^n$.

To show the induction step, we note that there must be a first time a production $Z \to C_iD_j$ is applied. Following this, as we previously observed, we will have a sentential form in

$$Z \Rightarrow (L^*(Z!F)R^*)^*W(L^*(Z!F)R^*)^* .$$

Now let us assume there are p applications of the productions in question to the left of the W and q to the right and p+q=n. From what we have seen before, this means the set of sentential forms we can generate is

$$\begin{aligned} &L^*(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^p R^*)^* W(L^*(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^q R^*)^* \\ &= ((L^*(Z!F)R^*)^*W)^p(L^*(Z!F)R^*)W(L^*(Z!F)R^*)^*(W(L^*(Z!F)R^*)^*)^q \\ &= ((L^*(Z!F)R^*)^*W)^{p+1}((L^*(Z!F)R^*)^*W)^q(L^*(Z!F)R^*)^* \\ &= ((L^*(Z!F)R^*)^*W)^{p+q+1}(L^*(Z!F)R^*)^* \end{aligned}$$

Hence the hypothesis holds.

Since we cannot bound the number of times productions of the form $Z \rightarrow C_iB_j$ will be used, we replace the exponent by a star. Adding the final production $Z \rightarrow \Lambda$, we then have that the set of words which Z can produce lie in the regular expression

$$((L^*(\Lambda!F)R^*)^*W)^*(L^*(\Lambda!F)R^*)^* .$$

We note that L's, F's, R's and W's can be computed in any quantity in any order; hence the regular event we derive is just

$$(L!F!R!W)^* .$$

*Example 2.*

The grammar associated with the subjecy/object take and grant system [12] is an example of a nondiscriminating grammar. Given the definition of the nonterminals shown in figure 6, it can be demonstrated that we have the grammar shown in figure 7. If we assume A is our starting symbol, we can eliminate productions 3, 4, 6, 8, 11, 12, 14 and 16, thereby giving us a nondiscriminating grammar.

Following the mechanical transformations used in the proof of the theorem, we see the regular expression associated with A is as follows:

$$(bd^*p \;!\; bd^*ph^*g \;!\; jh^*g \;!\; bd^*\ell h^*g \;!\; e \;!\; bd^*c \;!\; e \;!\; fh^*g \;!\; j \;!\; bd^*k \;!\; m \;!\; nh^*g)^*(a \;!\; bd^*c)$$

These methods provide us with a means for giving an alternative proof of the Theorem 2 in [12].

| | | | |
|---|---|---|---|
| $A = (S,\overrightarrow{r},S)$ | $B = (S,\overrightarrow{r},0)$ | $C = (0,\overrightarrow{r},S)$ | $D = (0,\overrightarrow{r},0)$ |
| $E = (S,\overleftarrow{r},S)$ | $F = (S,\overleftarrow{r},0)$ | $G = (0,\overleftarrow{r},S)$ | $H = (0,\overleftarrow{r},0)$ |
| $I = (S,\overrightarrow{w},S)$ | $J = (S,\overrightarrow{w},0)$ | $K = (0,\overrightarrow{w},S)$ | $L = (0,\overrightarrow{w},0)$ |
| $M = (S,\overleftarrow{w},S)$ | $N = (S,\overleftarrow{w},0)$ | $O = (0,\overleftarrow{w},S)$ | $P = (0,\overleftarrow{w},0)$ |

*Figure 17*

| | | | | | |
|---|---|---|---|---|---|
| 1. | $A \to ZR_a$ | $R_a \to a$ | $R_a \to R_b c$ | | |
| 2. | $B \to ZR_b$ | $R_b \to R_b d$ | $R_b \to b$ | | |
| 3. | $C \to OA$ | | | | |
| 4. | $D \to OB$ | | | | |
| 5. | $E \to L_e Z$ | $L_e \to e$ | $L_e \to fL_g$ | | |
| 6. | $F \to EJ$ | | | | |
| 7. | $G \to L_g Z$ | $L_g \to hL_g$ | $L_g \to g$ | | |
| 8. | $H \to GJ$ | | | | |
| 9. | $I \to ZR_i$ | $R_i \to i$ | $R_i \to R_b k$ | | |
| 10. | $J \to ZR_j$ | $R_j \to j$ | $R_j \to R_b \ell$ | | |
| 11. | $K \to OI$ | | | | |
| 12. | $L \to OJ$ | | | | |
| 13. | $M \to L_m Z$ | $L_m \to m$ | $L_m \to nL_g$ | | |
| 14. | $N \to MJ$ | | | | |
| 15. | $O \to L_o Z$ | $L_o \to g$ | $L_o \to pL_g$ | | |
| 16. | $P \to OJ$ | | | | |
| 17. | $Z \to ZZ$ | $Z \to A$ | $Z \to E$ | $Z \to I$ | $Z \to M$ |
| | $Z \to JG$ | $Z \to BO$ | $Z \to \Lambda$ | | |

*Figure 18*

## *Non-Grammatical Protection Systems*

As useful as the concept of grammatical protection systems is to obtaining linear time results to the safety question, a great many systems described in the literature fail to possess this property [2,5,13].

In this section we wish to show that certain systems, while failing to be truly grammatical, are sufficiently close to grammatical systems to enable us to utilize the results of the last section.

We will say a protection system is *near-grammatical* if for each right $\alpha$ there is some regular expression $E_\alpha$ such that a necessary condition for a vertex X to $\alpha$ a vertex Y is that they be connected by a path with word in $E_\alpha$; furthermore, this condition becomes sufficient if at certain identifiable points in the regular expression we check that certain more global conditions are satisfied. We assume these conditions do not involve the vertex x, and they can be verified in constant time (i.e., independent of the number of edges in the graph).

*Theorem 10:* The safety question for Near-Grammatical systems can be answered in linear time in the size of the protection graph.

*Proof:* This theorem is proved in a similar fashion to the previous Theorem 8. We place "finger" symbols in the places in the regular expressions where the conditions are to be verified. Again we assume to have a finite state automaton with T states and a protection graph with B vertices. Again we construct a new graph with B×T vertices, only this time we connect an arc from $(G_i,T_j)$ to $(G_k,T_l)$ *iff*

1) there was an arc from $G_i$ to $G_k$ in the original graph, and if we were in state $T_j$ at the point $G_i$ that arc would carry us to state $T_l$, or,

2) one of the "finger conditions" is true for $G_i$. In this case, k=i and $T_l$ is the state we would transfer to having accepted that "finger" in state $T_j$.

Again, having constructed the graph the result is then a standard reachability argument from automata theory.

*Example 3.*

In many current protection systems having a right to an object does not, as we have been assuming, automatically allow you to pass that right on to another individual. For instance, in the Multics system an individual can have access to a file only if his name is written on a list of individuals who are permitted to have that right. Therefore, if X has certain access privileges, another vertex Y, no matter what relationship it may have with X, cannot obtain those privileges without somehow getting its name on the list of permitted individuals.

We model this situation by means of a special right called *control* [2]. Having Control rights over X could, for instance, mean having the ability to write on the list of people who can access X.

We will use the subject-only take and grant transition rules of [12], only we include the concept of control. The control privilege cannot be passed. The rules are shown in figure 19.

That the system is not grammatical can be easily demonstrated. In the first graph in figure 20, X can obtain the read rights to Z, but it cannot do so in either of the two following graphs, thereby demonstrating that the ability to obtain rights does not depend solely upon the nature of the path between the two vertices.

We can observe that for $\alpha \in \{r,w\}$, X can obtain $\alpha$ rights to Y *iff*

1) X and Y are connected by a path in $(r!w)^*$ and
2) every vertex on that path has control rights to y.

This system is obviously near-grammatical. Hence, the safety question can be answered in linear time.

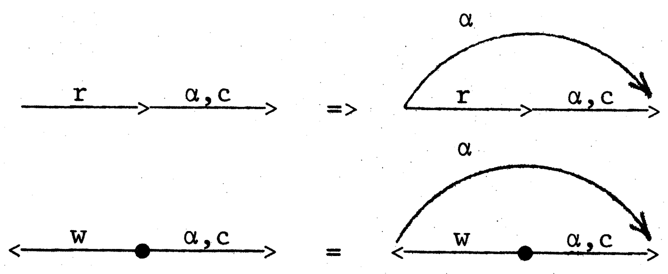


*Figure 19*

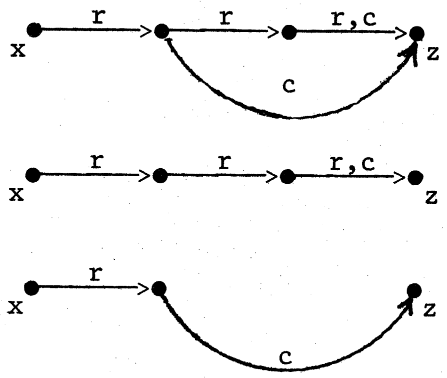


*Figure 20*

## *Conclusions*

The security of computer systems is a topic which appears will be of increasing concern in the near future. We feel that true understanding and trust in access privilege mechanisms which are proposed can only be achieved by formal analysis of the capabilities of these systems.

We have attempted to form a basis for the study of protection systems by classifying transformation rules which allow for formal analysis. In doing so we are trying to fill in the gap between a specific system for which linear time results can be demonstrated [12], and very general systems for which problems are known to be undecidable [7].

We hope that further research will bear out the utility of these studies by allowing us to model protection systems which are actually being used today. We feel the concept of a grammatical or near-grammatical system is natural and justified, since disregarding those systems which we are labeling "loose," if I have a right and I wish to give it to you I can only do so in a sense by passing it from hand to hand until it reaches you. Hence to a certain extent my ability to pass rights must depend upon the nature of the path between us.

It appears that further research along these lines will have important consequences not only for the formal analysis of abstract protection system models, but also for the practitioner who must design and implement actual access privilege mechanisms.

## *References*

[1] Aho, A. V., Hopcroft, J.E., and Ullman, J. D.
*The Design and Analysis of Computer Algorithms.*
Addison-Wesley, Reading, Mass., 1974.

[2] Bell, D. E., and LaPadula, L. J.
*Secure Computer Systems, Vol. I. Mathematical Foundations, Vol. II. A Mathematical Model.*
MITRE Corporation Technical Report MTR-2547, 1973.

[3] Dennis, J. B., and Van Horn, E. C.
Programming Semantics for Multiprogrammed Computations.
*Comm. ACM* 9,3 (March 1966) 143-155.

[4] Graham, G. S., and Denning, P. J.
Protection Principles and Practice.
*AFIPS Conference Proceedings* 40:417-429, 1972.

[5] Graham, R. M.
Protection in an Information Processing Utility.
*Comm. ACM* 11,5 (May 1968) 365-369.

[6] Harrison, M. A.
On Models of Protection in Operating Systems.
*4th Symposium on Mathematical Foundations of Computer Science* (1975).
Reprinted in *Lecture Notes in Comp. Science No. 32*, Springer-Verlag.

[7] Harrison, M. A., Ruzzo, W. L., and Ullman, J. D.
Protection in Operating Systems.
*Comm. ACM* 19,8 (August 1976) 461-471.

[8] Harrison, M. A.
Private communication.

[9] Hopcroft, J. E., and Ullman, J. D.
*Formal Languages and Their Relation to Automata.*
Addison-Wesley, Reading, Mass., 1969.

[10] Johnson, D. B.
*Algorithms for Shortest Paths.*
Ph.D. Thesis, Cornell University, 1973.

[11] Jones, A. K.
*Protection in Programmed Systems.*
Ph.D. Thesis, Carnegie-Mellon University, 1973.

[12] Jones, A. K., Lipton, R. J., and Snyder, L.
A Linear Time Algorithm for Deciding Security.
*Proc. 17th FOCS* (1976).

[13] Saltzer, J.H.
Protection and the Control of Information Sharing in MULTICS.
*Comm. ACM* 17,7 (July 1974) 388-402.

[14] Tsichritzis, D.
Protection in Operating Systems.
*Infor. Proc. Letters* 1 (1972) 127-131.

[15] Valiant, Leslie G.
General Context-free Recognition in Less than Cubic Time.
*JCSS* 10 (1975) 305-315.