THE TRANSFER OF INFORMATION AND AUTHORITY IN A PROTECTION SYSTEM‡

Matt Bishop‡‡

Lawrence Snyder‡‡‡

Department of Computer Science
Yale University
New Haven, Connecticut
06520

Abstract: In the context of a capability-based protection system, the term "transfer" is used (here) to refer to the situation where a user receives information when he does not initially have a direct "right" to it. Two transfer methods are identified: *de jure* transfer refers to the case when the user acquires the direct authority to read the information; *de facto* transfer refers to the case when the user acquires the information (usually in the form of a copy and with the assistance of others), without necessarily being able to get the direct authority to read the information. The Take-Grant Protection Model, which already models *de jure* transfers, is extended with four rewriting rules to model *de facto* transfer. The configurations under which *de facto* transfer can arise are characterized. Considerable motivational discussion is included.

1. *Introduction*

Recall that a capability-based protection system [1,2] is a mechanism which limits access to information to those entities, e.g. users, that have the "right" to access it. The "rights" are usually represented as tokens and the system keeps a complete record of which entities have which "rights." All references to information are validated to insure that only rightful access is permitted.

In this context there are two distinct means by which a user that does not have the "right" to read a file can acquire the information. The first will be called* *de jure* acquisition:

> *de jure* acquisition means that the authority or "right" to read the information is transferred to the user.

For example, another user that has the read "right" can grant it to the user whereupon he invokes the "right" and reads the information. Of course, it may be necessary to "pass" the "right" along through several users before it can reach the end user.

The second method is called *de facto* acquisition:

> *de facto* acquisition means that the user acquires the "information" in the file without necessarily acquiring the direct authority to access the file.

For example, the user may be given a *copy* of the file from another user who does have the "right" to read it. The copy could be passed via a "mailbox" file common to both users. It may be necessary for several users to pass copies of the file along. We may also include other methods of transfer such as a user writing directly into another user's address space. Lampson considered some of these issues in his discussion of the confinement problem [4].

These acquisitions are illustrated diagrammatically in Figures 1 and 2. The letters represent "rights" where r,w and g abbreviate read, write and grant, respectively. In Figure 1, the protection state is changed when Baker performs a *de jure* transfer by granting Abel the read "right" to File 2. Figure 2 illustrates that there is a potential for a *de facto* transfer of File 3 to Baker since Charlie could copy the contents of File 3 into the "mailbox." A dashed line is used to emphasize that *no change* in the protection state has taken place -- we have only illustrated one potential *de facto* transfer.

Although *de jure* and *de facto* transfers accomplish the same objectives -- movement of

----

*Our use of *de jure*, "rightful, by right" [3], and *de facto*, "(existing) in fact, whether by right or not" [3], is intended to avoid pejorative names such as authorized/unauthorized, legal/nonlegal, etc.

----

‡This research was sponsored in part by National Science Foundation Grant MCS77-12517 and Office of Naval Research Grant N00014-75-C-0752. Preparation of this report was facilitated by the Theory Net, National Science Foundation Grant MCS78-01689.
‡‡Present address: Department of Computer Science, Purdue University, West Lafayette, IN 47907
‡‡‡Present address: Department of Computer Science, University of Washington, Seattle, WA 98195

information* -- they are quite different.

- *De jure* transfer implies *de facto* transfer, but not vice versa. That is, a *de jure* transfer can be thought of as a *de facto* transfer where the authority happens to be acquired. But the recipient of a *de facto* transfer may not be able to get the "read rights" to the source version of the file and thus no *de jure* transfer is possible.
- *De jure* transfer always gives the right to obtain an up-to-date version of the information whereas a *de facto* transfer will not reflect any changes made to the original file since the copy was created. This means that *de facto* transfer is inferior to *de jure* transfer for frequently changed files.
- *De facto* transfer relies on the agents transmitting a complete and accurate copy of the file. Any errors introduced either intentionally or accidentally during transmission may degrade the "quality" of the information.

Finally, and perhaps most importantly, currently available formal models of capability-based protection systems [5,6,7] have focused only on *de jure* transfer -- *de facto* transfers have not been studied in the context of these models. The purpose of this paper is to present a formal model to aid in understanding *de facto* information transfers.

Since *de jure* transfers may assist in accomplishing a *de facto* transfer (Figure 3), it will be helpful to cast our study of *de facto* transfers in a context where *de jure* transfers are already understood. The Take-Grant Model [6] is appropriate for these purposes and so we shall extend it to include *de facto* transfers as well as *de jure* transfers. (No knowledge of the Take-Grant Model is presumed; this paper is self-contained.)

Our plan is to give in Section 2 a separate model (compatible with the Take-Grant Model) of just unaided *de facto* transfers. After a brief philosophical discussion in Section 3, we proceed in Section 4 with a characterization of potential *de facto* transfers. In Section 5 we review known results concerning *de facto* transfers. Then, in Section 6 we permit both types of transfer and characterize the general *de facto* transfer case. We present conclusions and open problems in Section 7.

2. *De Facto* Information Transfers

A capability-based protection system will be modeled by a finite, directed graph called a *protection graph*, analogous to the protection graphs used in earlier versions of the Take-Grant Model [6]. The protection graph is intended to abstract the protection state of the system, i.e. that information recorded in the protection system concerning which entities have which "rights" to other entities.

The vertices of the graph will be of two types: *subjects* (denoted by ●) will represent "active" entities such as users, and *objects* (denoted by o) will denote "passive" entities such as files. (There are usually many other entities in a system, e.g. load modules, directories, etc., that are hard to categorize by such vague terms as "active" or "passive." For example, one might argue that a load module is "active" in the sense that it could, when executed, cause information to move. Alternatively, if one knows that the module is "secure," i.e. doesn't disseminate information, it might be called "passive." These and other interpretations depend upon the particular system being modeled, and because of our general approach, they are beyond the scope of this study. We simply provide two classes of entities and depend on the reader to make the appropriate classification for his system.)

The edges between the vertices are labeled with elements from a finite alphabet R corresponding to "rights." In order to develop our theory, we assume that R contains the letters r,w,t,g mnemonic for *r*ead, *w*rite, *t*ake and *g*rant. The interpretation of an edge from vertex x to vertex y labeled by $\alpha \subseteq R$

$$\underset{x}{\bullet}\xrightarrow{\alpha}\underset{y}{\circ}$$

is that within the protection system, x has all and only the α rights to y. We call these edges *explicit* since they represent authority that is formally recorded in the protection system. Clearly, every *de jure* transfer will cause a change in the protection graph. But when we identify the potential for a *de facto* acquisition by user x of the information in file y we cannot indicate this fact by adding an explicit x-to-y edge labeled r in the protection graph. This is because the protection graph records the authority relationships and a potential *de facto* transfer, regardless of whether it occurs, does not change the authority. Thus, we permit a second kind of edge, labeled with an r and denoted by a dashed line that will represent potential *de facto* acquisition. These edges will

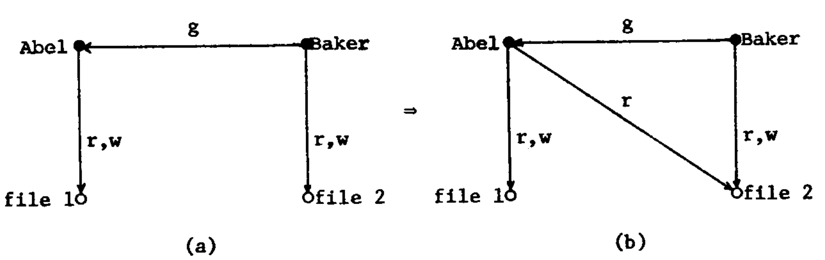


Figure 1: A *de jure* acquisition: Baker grants to Abel the read authority to File 2.

*Our use of the phrase "movement of information" should not be confused with "information flow" as used in, say, [11,12]

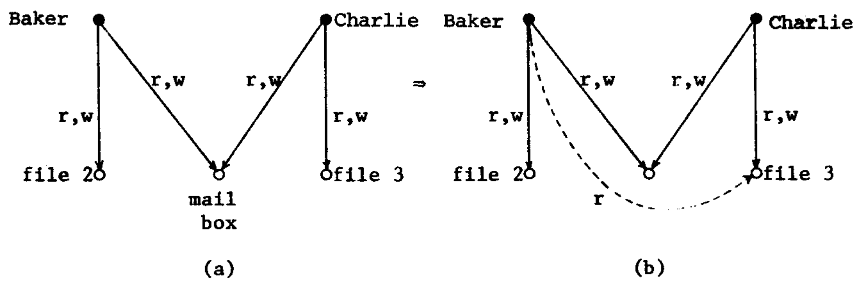
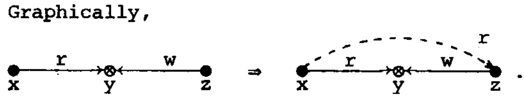


Figure 2: A potential *de facto* acquisition: Charlie could copy file 3 into the mailbox from which Baker could read the information.

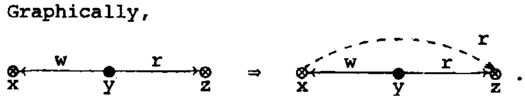
be called *implicit* edges.

With the protection state thus abstracted as a directed graph, it remains to explain how to identify potential *de facto* acquisitions. In order to be compatible with the Take-Grant Model we present four graph rewriting rules that abstract some of the basic methods of *de facto* information acquisition. (Note in the following definitions, the unmodified use of "edge" refers to either implicit or explicit edges. In the diagrams, ⊗ denotes a vertex that can be either a subject or an object and set braces are elided.)

Post: Let x, y and z be distinct vertices in a protection graph G such that x and z are subjects. Let there be an edge from x to y labeled α, $r \in \alpha$, and an edge from z to y labeled β, $w \in \beta$. Then *post* defines a new graph G' with an implicit edge from x to z labeled {r}. Graphically,

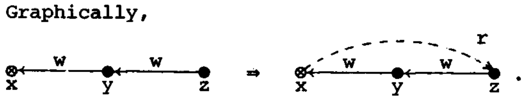


Pass: Let x, y and z be distinct vertices in a protection graph G such that y is a subject. Let there be an edge from y to x labeled by α, $w \in \alpha$, and an edge from y to z labeled by β, $r \in \beta$. Then *pass* defines a new graph G' with an implicit edge from x to z labeled {r}. Graphically,



Spy: Let x, y and z be distinct vertices in a protection graph G such that x and y are subjects. Let there be an edge from x to y labeled α, $r \in \alpha$, and an edge from y to z labeled β, $r \in \beta$. Then the *spy* rule defines a new graph G' with an implicit edge from x to z labeled {r}. Graphically,

Find: Let x, y and z be distinct vertices in a protection graph G such that y and z are subjects. Let there be an edge from y to x labeled α, $w \in \alpha$, and an edge from z to y labeled β, $w \in \beta$. Then *find* defines a new graph G' with an implicit edge from x to z labeled {r}. Graphically,

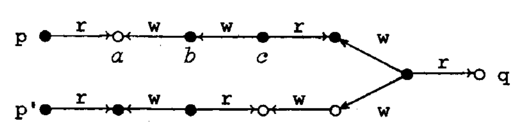


We will refer to these rules, collectively, as the DF rules.

The rules are intended to abstract possible ways in which information may be read by vertex x with the cooperative effort of one or more subjects. The subjects invoke authority that they own within the system in order to effect *de facto* transfer. This transfer, or more accurately, the potential for this transfer, is summarized by the implicit edge from x to z, labeled {r}. We can then apply these rules to a protection graph (see example below) to summarize the *de facto* transfer in the entire system.

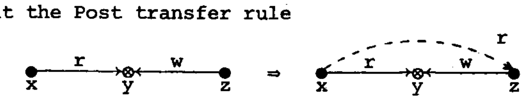
The Post rule abstracts parts of the operation described in the Introduction of transfer via a mailbox. In the Pass rule y acts as a conduit through which information travels from z to x. The Spy rule abstracts the case where y reads information from z and x "watches" y read the information. More often, however, it is used to "compose" transfers (see graph G_4 in the example below). The Find rule abstracts the case where z deposits data into the address space of y and y in turn deposits it into x.

The rewriting rules enable us to illustrate the potential *de facto* transfers by augmenting a given protection graph G with new implicit edges. Let G_0 be the protection graph

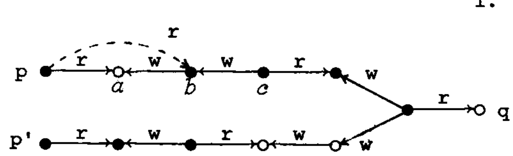


G_0

and consider whether or not p can read q. We note that the Post transfer rule



matches so it can be applied where the variables of the rule definition (x,y,z,α and β) match p,*a*,*b*,{r} and {w}, respectively. Thus, we summarize the potential for this transfer by adding an implicit edge from p to *b* labeled r. The result is G_1.



G_1

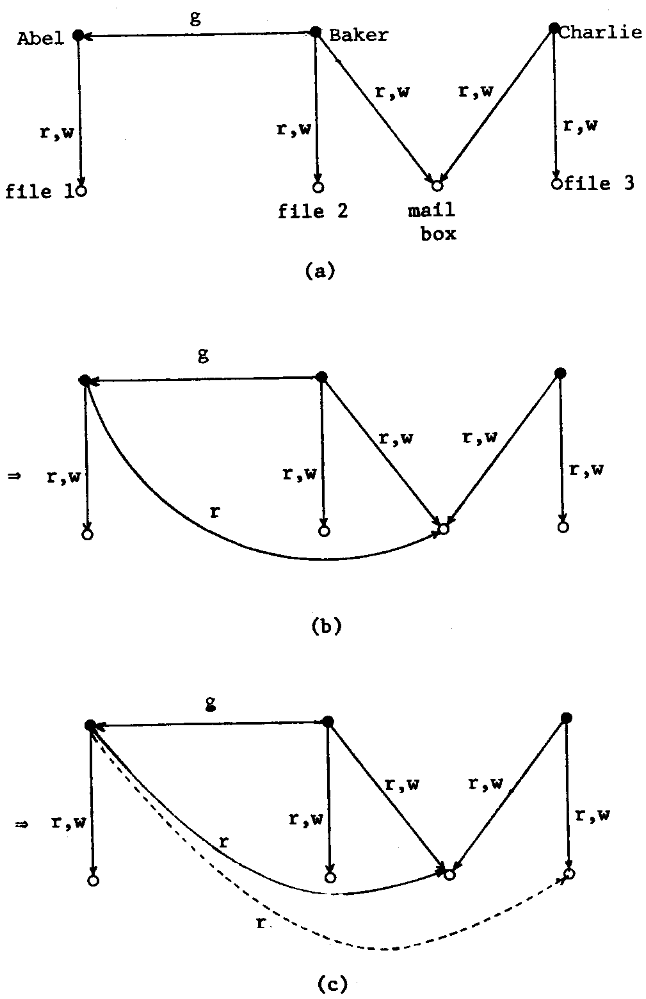
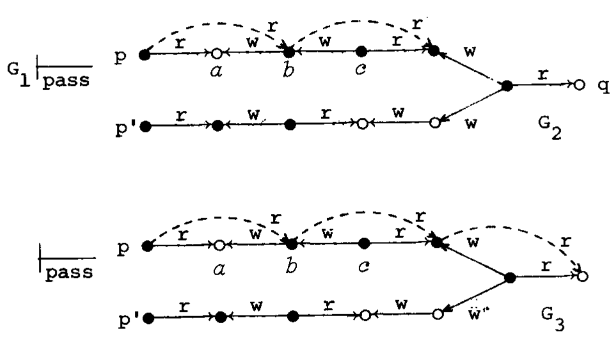
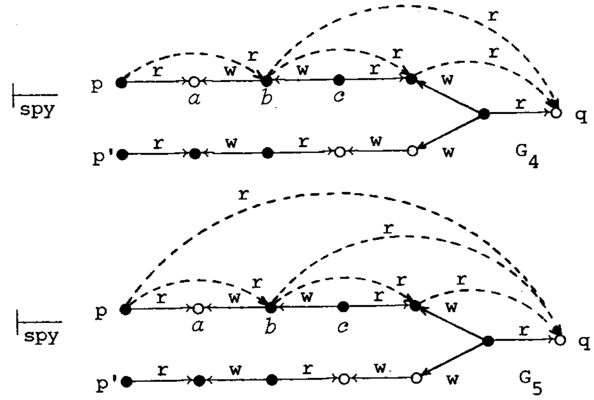


Figure 3: Given the protection state (a), a *de jure* transfer (b) is used to enable a *de facto* transfer (c).

Usually, we denote such a rule application by $G_0 \vdash_{post} G_1$, and read "G_0 yields G_1 via Post."

The sequence of rule applications that illustrate that p could acquire the contents of q are illustrated below.

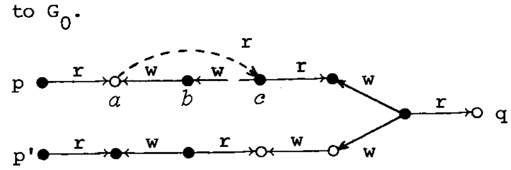




So we conclude that there is a potential for *de facto* transfer to p. Note that all of these added edges are *implicit* -- they do not represent added authority, only potential *de facto* access.

Tortuous though the example may be, it illustrates that rather complex transfer can be realized. It is just as important (perhaps more important) to know what *de facto* transfers cannot be realized. For example, it is *not* possible for p' to read q by a transfer along the "lower" path in G_0. This is because of the two consecutive objects which form a "barrier" to indirect transfer. (See Theorem 4.1.)

To illustrate another subtlety, note that *b* plays a pivotal role in the transfer. We might have tried to skip past *b* by applying the Find rule to G_0.



But *a* is an object, and our rule definitions do not permit the application of a Spy to define a read edge from p to *c*. One might argue that a Spy should be allowed here because the *a*-to-*c* read edge is implicit and thus *a* receives the information passively. Subjecthood appears restrictive. Our decision to force the second vertex in a Spy rule to be a subject guarantees the existence of an agent when needed. It will be clear from our results that this limitation is not serious.

3. *Discussion*

We have introduced a considerable amount of mechanism with only the briefest motivational remarks. It is appropriate to pause and discuss what we have and have not done, and why.

We emphasize that we are basing our study of *de facto* transfer on the protection graph -- an abstraction of the authority relations in the system. Since *de facto* transfers involve the movement of information rather than authority, one cannot tell by observing the protection graph whether or not a *de facto* transfer has occurred. One can only tell if it *could* occur in any particular arrangement of authority relations. Accordingly, we have spoken of "potential *de facto* transfer."

It may seem curious that the protection graph is used for studying *de facto* transfers when it doesn't even provide a means of witnessing such transfers. But there appears to be little alternative. First, *de facto* transfers must operate within the extant authority relations of the system together with those changes that could be achieved via *de jure* transfers. Since the *de jure* transfers are visible in the protection graph, it is needed to understand those changes. Second, it appears unlikely that we will ever be able to determine when or if a *de facto* transfer has occurred. Observing that some bits have been transmitted is not enough. For example, in Figure 2 when Charlie copies "information" from File 3 into the mailbox and Baker reads the contents of the mailbox, does Baker receive "information?" Certainly, if Charlie exactly copies File 3, the answer is probably "yes." The answer should also be "yes" if Charlie copies a scrambled version of File 3 that Baker can decode, but this would be hard to distinguish from the case where Charlie only transmits garbage. If Charlie only reports that File 3 "is an ASCII file with 107,261 bits, 26,889 of which are 1," then it is unlikely, though not impossible that this is "information" to Baker. And if Charlie reports that "File 3, if it exists, contains the contents of File 3, if any" then no "information" is transferred, provided this response is not some clever encoding scheme. In short, "information" transfer has very little to do with transmitting bits and very much to do with semantic and logical issues that are well beyond the state-of-the-art.

We regard the four DF rules defined in the previous section as a representative sample of the potential *de facto* acquisitions that might arise in a protection system. In some actual systems only a subset of these methods might be achievable. In others, there may be acquisitions not captured by these rules, e.g. if there is an explicit "update right." In either case, the development that follows may have to be modified. Our purpose in this paper is to illustrate the methodology used to access the potential *de facto* transfers of a protection system.

To be explicit, this methodology is to define a set of rewriting rules that add implicit edges to a protection graph and then to characterize (Sections 4-6) the conditions under which these rules permit *de facto* transfer.

As has already been noted, this methodology has the advantage that it is compatible with (and thus allows us to build upon) the Take-Grant Model. A more compelling advantage, perhaps, is that our "three-place" rewriting rule schemata, i.e. three vertices and two connecting edges, encapsulate all of the required components of an "information transfer."* In particular, if we distinguish between the *conveyance* of information (i.e. reading the information when the read authority exists) and *transfer* of information (i.e. reading the information when the read authority doesn't originally exist), then a transfer obviously requires (1) an information source, (2) a receiver, (3) at least one agent, since the information and receiver aren't presumed to be connected, (4) a "right" or a chain of "rights" connecting the agent to the information and (5) a "right" or a chain of "rights" connecting the agent to the receiver. In our rules the "middle vertex," y, is the agent. It plays either an active or a passive role. The other four components of a transfer are also explicitly represented.

Our approach has explicitly abstracted the notion of *de facto* transfer as the composition of two conveyance operations. But it has come to our attention [8,9] that an alternative approach can be based on conveyance alone. The suggestion [9] is to use "two place" rules, i.e. two vertices connected by an edge, that describe the circumstances under which a "token" (corresponding to the information) can be moved along an edge from one vertex to another. Although this approach does not fulfill our original goal of explicitly abstracting *de facto* information transfer, it does have an appealing technical simplicity. Whether this benefit is offset by difficulties in interfacing with *de jure* transfer remains an open question that will not be pursued here.

Finally, we must make one cautionary remark concerning the interpretation of protection graphs. This is a general study that will be applicable (we hope) to a wide class of protection systems. As such we must consider all protection graphs even if they do not have a sensible interpretation in the context of a *particular* protection system. For example, we allow such constructs as $x \xrightarrow{r} y$ in our protection graphs. If one interprets objects as files, this may be meaningless. But if objects include "secure" processes, then this is more reasonable. We cannot limit *a priori* the class of interpretations, so we allow for any protection graph consistent with our original definitions.

---

*We are not trying to split hairs over the semantics of "conveyance" and "transfer"; we are only assigning suggestive names to the two concepts.

4. *The Conditions of* De Facto *Transfer*

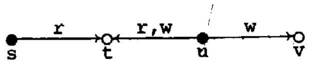
Having abstracted potential *de facto* transfers as a set of four rewriting rules and having illustrated that these rules compose in complex ways, we now formulate an exact statement of what it means for a potential *de facto* transfer to exist within the model. This will be done by defining a predicate *can•know•f*(p,q,G) of three parameters. The predicate is true if vertex p of protection graph G can acquire the information from vertex q of G by some sequence of rule applications. Then, we define conditions on G that determine when the predicate is true.

Define for a protection graph G_0 and arbitrary distinct vertices p and q of G_0

> *can•know•f*(p,q,G_0) to be true if and only if
>
> there exists a sequence of graphs $G_1,\ldots,G_n$ ($n \geq 0$) such that G_{i+1} follows from G_i ($0 \leq i < n$) by one of the DF rules and in G_n either a p-to-q edge labeled r exists or a q-to-p edge labeled w exists and if the edge is explicit, its source is a subject.

Thus, the predicate *can•know•f*(p,q,G_0) is true if and only if the authority already exists in G_0 or an implicit edge from p-to-q can be added by means of the four DF rules.

Now, we formulate conditions under which *can•know•f* holds. To aid in this endeavor, define an *rw-path* in a protection graph G as a sequence of vertices $v_0, v_1, \ldots, v_k$ ($k \geq 1$) such that v_i is connected to v_{i+1} by an edge (in either direction) labeled with r or w (or both) for all i, $0 \leq i < k$. We say that the rw-path is *between* v_0 and v_k. For example, in the graph



the sequence s,t,u,v is an rw-path.

Not all rw-paths will permit *de facto* transfer of information. (For example, s,t,u,v above does not!) So we limit our attention to a certain subset of them. To do this, we associate with each rw-path one or more words over the alphabet $\{\vec{r}, \overleftarrow{r}, \vec{w}, \overleftarrow{w}\}$ in the obvious way; for example, the sequence s,t,u,v given above has two associated words, namely $\vec{r}\overleftarrow{r}\vec{w}$ and $\vec{r}\overleftarrow{w}\vec{w}$.

Define an rw-path $v_0, v_1, \ldots, v_k$ ($k \geq 1$) to be an *admissible rw-path* if and only if

(i) it has an associated word $a_1 a_2 \ldots a_k$ in the regular language $(\vec{r} \cup \overleftarrow{w})^*$ and

(ii) if $a_i = \vec{r}$ then v_{i-1} is a subject and if $a_i = \overleftarrow{w}$ then v_i is a subject.

There are two immediate consequences of this definition. First, since $k \geq 1$, there is always at least one letter in the word associated with any admissible path. Second, there cannot be two consecutive objects on any admissible path.

The first result concerning *de facto* transfers can now be stated.

Theorem 4.1: Let p and q be vertices in a protection graph G. Then *can•know•f*(p,q,G) is true if and only if there is an admissible rw-path between p and q.

The proof of this result is a rather routine induction and a closely related variant of the proof can be found in [10]. We emphasize that this condition is both necessary and sufficient; it exactly characterizes the *de facto* transfers in the entire system.

In the definition of rw-path we permitted cycles. It is easy to prove that these cycles are redundant, i.e. an admissible path with no cycles can be found from any admissible path by "snipping off" the cycles. Thus, it is easy to test the conditions of the theorem for any pair of vertices by using standard breadth-first graph traversal techniques.

Corollary 4.2: For vertices p and q of a protection graph G, there is a linear-time (in vertices plus edges), algorithm for testing *can•know•f*(p,q,G).

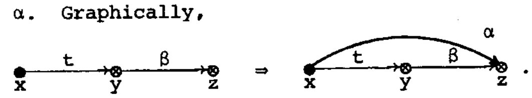
The reader is encouraged to return to the graph G_0 in Section 2 to verify our claim that there can be no transfer along the "lower" path; that is, *can•know•f*(p',q,G_0) is false.

5. *Review of* De Jure *Transfer*

Up to this point we have concentrated on the four rules that implement *de facto* transfers. Although these rules specify the addition of an edge in the graph, we have agreed that these are only *implied* edges -- no new access authority has been created. Now, we review the way in which *de jure* acquisition takes place in the Take/Grant Model.

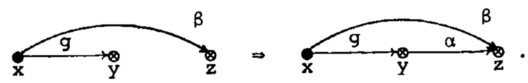
Recall that in addition to r and w, there are two other rights: t and g. In [6] the following rules were introduced for changing access authority.* All edges referred to in these rules are explicit.

Take: Let x, y and z be three distinct vertices in a protection graph G such that x is a subject. Let there be an explicit edge from x to y labeled γ such that $t \in \gamma$, an explicit edge from y to z labeled β and $\alpha \subseteq \beta$. Then the *take* rule defines a new graph G' by adding an explicit edge to the protection graph from x to z labeled α. Graphically,



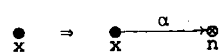
The rule can be read: "x takes (α to z) from y."

Grant: Let x, y and z be three distinct vertices in a protection graph G such that x is a subject. Let there be an explicit edge from x to y labeled γ such that $g \in \gamma$, an explicit edge from x to z labeled β, and $\alpha \subseteq \beta$. The *grant* rule defines a new graph G' by adding an explicit edge from y to z labeled α. Graphically,



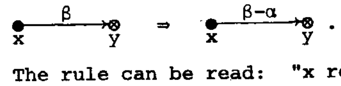
The rule can be read: "x grants (α to z) to y."

Create: Let x be any subject vertex in a protection graph G and let α be a subset of R. *Create* defines a new graph G' by adding a new vertex n to the graph and an explicit edge from x to n labeled α. Graphically,



The rule can be read: "x creates (α to) new {subject / object} n."

Remove: Let x and y be any distinct vertices in a protection graph G such that x is a subject. Let there be an explicit edge from x to y labeled β, and let α be any subset of rights. Then *remove* defines a new graph G' by deleting the α labels from β. If β becomes empty as a result, the edge itself is deleted. Graphically,



The rule can be read: "x removes (α to) y."

We refer to these four rules collectively as the *DJ rules*.

The edges added by these rules represent explicit changes in the access authority. Thus, when "x takes (r to z) from y," x only acquires the read

---

*In [6] these rules were labeled with different letters.

rights to the information. It must invoke the right to read the information. In addition to adding edges, Create allows the addition of new vertices. As Figure 4 illustrates,* Create adds an important dimension to the model since without Create one cannot add g to the *a-to-b* edge in this example.

In order to report on previous results [6,7] we define *tg-path* (analogous to an rw-path) as a nonempty sequence $v_0,\ldots,v_k$ of vertices such that for all i, $0 \le i < k$, v_i is connected to v_{i+1} by an edge (in either direction) with a label containing a "t" or "g" (or both). Vertices are *tg-connected* if there is a tg-path between them and we call any maximal, tg-connected subject-only subgraph an *island*.

Associate with tg-paths words over the alphabet $\{\vec{t},\overleftarrow{t},\vec{g},\overleftarrow{g}\}$ analogous to the words associated with rw-paths. (If k = 0 in the tg-path, then the associated word is ϵ.) A tg-path $v_0,\ldots,v_k$ with v_0 being a subject is an *initial span* if it has an associated word in the language $\{\vec{t}^*\vec{g}\} \cup \{\epsilon\}$; it is a *terminal span* if it has an associated word in $\{\vec{t}^*\}$; and it is a *bridge* if v_k is a subject and it has an associated word in $\{\vec{t}^*, \overleftarrow{t}^*, \vec{t}^*\vec{g}\overleftarrow{t}^*, \vec{t}^*\overleftarrow{g}\overleftarrow{t}^*\}$. Note that initial and terminal spans have orientation, i.e. v_0 is the *source* of the spans. We say v_0 initially or terminally spans to v_k.

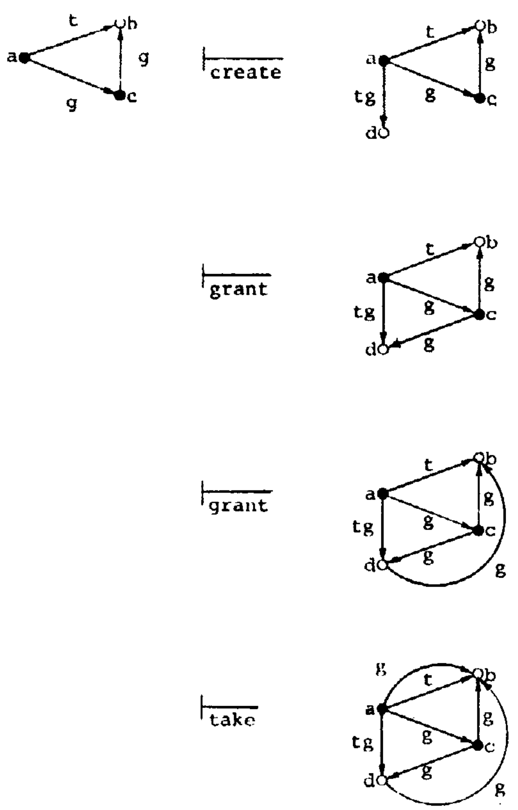


Figure 4: Vertex a acquires g rights to b, i.e. g is added to the label on the a-to-b edge. The rule applications may be read:

a creates (tg to) new object d,

a grants (g to d) to c,

c grants (g to b) to d,

a takes (g to b) from d.

Restricting our attention only to Take, Grant, Create and Remove, we define for a right α and distinct vertices p and q of a protection graph G_0, the predicate

can·share$(\alpha,p,q,G_0) \Leftrightarrow$ there are protection graphs $G_1,\ldots,G_n$ such that $G_0 \vdash^* G_n$ using only DJ rules and in G_n there is a p-to-q edge labeled α.

Note that α can be any right in R, including $\{r,w,t,g\}$.

We may now state when the *can·share* predicate is true. Let p and q be arbitrary, distinct vertices in protection graph G_0 and let $\alpha \in R$.

Theorem 5.1[6]: The predicate *can·share*(α,p,q,G_0) is true if and only if the following hold simultaneously:

(i) there is a vertex $s \in G_0$ with an s-to-q edge labeled α,

(ii) there exist subject vertices p' and s' such that

(a) p' initially spans to p,
(b) s' terminally spans to s,

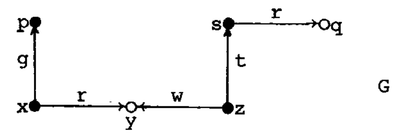
(iii) there exist islands $I_1,\ldots,I_v$, $p' \in I_1$, $s' \in I_v$, and there is a bridge from I_j to I_{j+1} $(1 \le j < v)$.

Figure 5 illustrates the conditions of the theorem. Although these conditions appear to be complicated, we can test a protection graph in linear time to see if it satisfies the conditions.

Clearly, if one is restricted to the DJ rules, then p can get *de jure* access to q in G_0 if and only if *can·share*(r,p,q,G_0) is true. The crucial question is: how do the DJ and DF rules interact? We describe that in the next section.

6. *Combined Transfers*

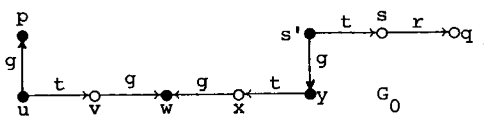
We begin by illustrating a simple case where both *de jure* and *de facto* transfers are needed to share information. Consider the protection graph G:



and notice that *can·share*(r,p,q,G) is false since s (the only owner of the "read right" to q) is not tg-connected to p. Also, *can·know·f*(p,q,G) is false

---

*Note, even though there is only one directed edge from any vertex *a* to any vertex *b*, we occasionally draw two to emphasize changes in labelling.



Islands: $I_1 = \{p,u\}$, $I_2 = \{w\}$, $I_3 = \{y,s'\}$.

Bridges: u,v,w and w,x,y.

Initial span: p; associated word: ϵ.

Terminal span: s',s; associated word: $\vec{t}$.

Can•share(r,p,q,G_0) is true as the following derivatives attest:

1. s' takes (r to q) from s.
2. s' grants (r to q) to y.
3. y takes (g to w) from x.
4. u takes (g to w) from v.
5. u grants (g to p) to w.
6. y grants (r to q) to w.
7. w grants (r to q) to p.

The resulting graph appears as follows:

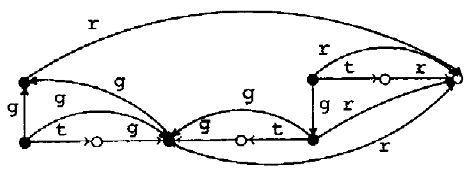
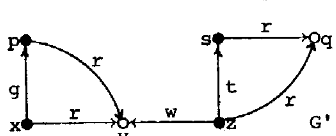


Figure 5: Illustration of the conditions of *can•share*.

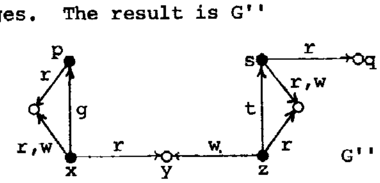
---

since there is no admissible rw-path between p and q. Furthermore, by our Theorem 5.1, no matter what changes we make to G using Take, Grant, Create and Remove alone, *can•share*(r,p,q,G) remains false, and by our Theorem 4.1 no matter what changes we make to G using Spy, Post, Pass and Find alone, *can•know•f*(p,q,G) remains false. But it *is* possible using DJ and DF rules to construct a graph G' in which *can•know•f*(p,q,G) is true.

In fact, there are two ways to change the graph that are conceptually different. First, x can grant (r to y) to p and z can take (r to q) from s. This results in graph G'.



which now contains an admissible rw-path. Alternatively, in G vertices x and s can create r,w rights to new objects and "read rights" to these objects can be acquired by p and z to "straddle" the t and g edges. The result is G''



which contains an admissible rw-path. Thus, we can either transmit existing rights or create new rights to build an rw-path.

We refer to the use of any combination of the DJ and DF rules as *combined transfer*, even though it is implementing a *de facto* transfer. (Recall that the DJ rules can only match explicit edges while the DF rules can match explicit or implicit edges.)

Following our paradigm, we define a predicate that introduces an "r" edge by any of the combined transfers. Let p and q be arbitrary, distinct vertices in a protection graph G_0, then

> *can•know*(p,q,G_0) is true if and only if there is a sequence of protection graphs $G_1,\dots,G_n$ such that $G_0 \vdash^* G_n$ and in G_n either a p-to-q edge labeled r exists, or a q-to-p edge labeled w exists and if the edge is explicit, its source is a subject.

Note that *can•know*(p,q,G_0) is simply *can•know•f*(p,q,G_0) without restrictions on the rule types.

Define *rwtg-path* in the obvious way and associate words over the alphabet $\{\vec{t},\overleftarrow{t},\vec{g},\overleftarrow{g},\vec{r},\overleftarrow{r},\vec{w},\overleftarrow{w}\}$ as usual. We define a second class of spans. Let $v_0,\dots,v_k$ (k>0) be an rwtg-path where v_0 is a subject. This path is an *rw-initial span* if its associated word is in the regular language $\{\vec{t}^*\vec{w}\}$ and it is an *rw-terminal span* if its associated word is in $\{\vec{t}^*\vec{r}\}$. Again we observe that spans have orientation and we say that v_0 *rw-initially* (or *rw-terminally*) spans to v_k.

Define the regular lanuages:

Bridges: $B = \{\vec{t}^* \cup \overleftarrow{t}^* \cup \vec{t}^*\vec{g}\,\overleftarrow{t}^* \cup \vec{t}^*\overleftarrow{g}\,\overleftarrow{t}^*\}$,

Connections: $C = \{\vec{t}^*\vec{r} \cup \overleftarrow{w}\,\overleftarrow{t}^* \cup \vec{t}^*\vec{r}\,\overleftarrow{w}\,\overleftarrow{t}^*\}$.

Note that the bridges language is the same set defined in Section 5.

We can now characterize the *can•know* predicate. Let p and q be arbitrary, distinct vertices in a protection graph G.

> *Theorem 6.1:* *can•know*(p,q,G) is true if and only if there exists a sequence of subjects $u_1,\dots,u_n$ in G ($n \geq 1$) such that the following conditions hold:
>
> (a) $p = u_1$ or u_1 rw-initially spans to p,
>
> (b) $q = u_n$ or u_n rw-terminally spans to q, and
>
> (c) for all i, $1 \leq i < n$ there is an rwtg-path between u_i and u_{i+1} with an associated word in $B \cup C$.

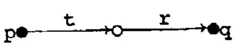
Although the proof is quite detailed, we can easily outline the overall flow of the argument. Showing that *can•know*(p,q,G) implies the conditions involves separating the rule applications of a witness into two classes: (i) those using only DJ rules, (ii) those using both DJ and DF rules. These latter types of applications are reordered so that (a) Creates are performed first, (b) the other DJ rules are performed next and (c) the DF rules are performed last. Then we observe that before the step (c) rule applications begin, an admissible rw-path must exist. The subjects of the admissible path are shown to satisfy the required conditions in the original graph, provided they exist in the original graph. If they do not exist, surrogates for them are found in the original graph and the surrogates are shown to have the required properties. The converse of the theorem is proved constructively. Details can be found in [10].

Although the proof is quite involved, the conditions are quite straight-forward. The reader is encouraged to return to the graph presented at the beginning of the section to verify that they do apply. Also, with some careful study of the set B ∪ C we can prove the following corollary.

> *Corollary 6.2:* For arbitrary, distinct vertices p and q in a protection graph G, the predicate *can•know*(p,q,G) can be tested in linear time in the size of the graph.

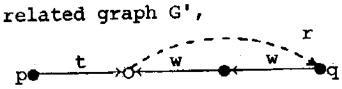
7. *Concluding Remarks*

As the reader reviews the conditions of Theorem 6.1 for *de facto* transfer, he will note that in the graph G,



G

the predicate *can•know*(p,q,G) is true, since a witness can be found by applying Take. But in the related graph G',



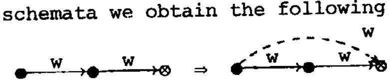
G'

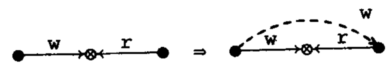
can•know(p,q,G') is false, since DJ rules cannot be applied to implicit edges. Is this inconsistent?

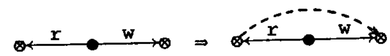
Not at all. In fact the similarity of the two graphs is based simply on the fact that *de facto* transfers are being recorded by "r" labeled edges. If we remove the implicit edge, the graphical similarity is lost. One might continue to argue, though, by observing that "information" can be "deposited" in the object and p should be allowed to take it. But Take isn't an operation on information, it is an operation on "rights" and the object has no rights to be Taken. Moreover, there is no edge labeled "r" from p to the object. So what appeared to be an inconsistency turns out to be quite consistent with our proposed interpretation. We should emphasize that even if inconsistencies do arise in this particular development, the methodology could be reapplied to a different set of rules to realize a more pleasing formulation.

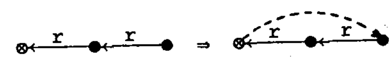
In the foregoing sections we have concerned ourselves with *de facto* transfers in which p can receive information from q -- a one-way transfer. Suppose p would like to communicate back to q, i.e. establish two-way communication. Must we repeat this entire development for the write right? Not at all.

Observe that by interchanging the r and w label on our DF rule schemata we obtain the following:

spy-w 

post-w 

pass-w 

find-w 

These new DF rules reflect the symmetry of read and write and are intuitively consistent.* Moreover, the directionality of the edges and the subject/object distinctions are all preserved. Thus, by interchanging r and w in the foregoing section, all substantive aspects of the arguments are preserved.

To emphasize this symmetry, define for arbitrary, distinct vertices p and q of a protection graph G

> *can•tell*(p,q,G) to be true if and only if there is a sequence of protection graphs $G_1,\ldots,G_n$ such that G_{i+1} follows from G_i by application of one of these new rules or the DJ rules ($0 \le i < n$) and in G_n
>
> a p-to-q edge labeled w exists or a q-to-p edge labeled r exists and if the edge is explicit the source is a subject.

Then we have from Theorem 6.1:

> *Corollary 7.1:* *can•tell*(p,q,G) is true if and only if there exists a sequence of subjects $u_1,\ldots,u_n$ in G ($n \ge 1$) such that the following conditions hold:
>
> (a) $p = u_1$ or u_1 wr-initially spans to p,
>
> (b) $q = u_n$ or u_n wr-terminally spans to q, and
>
> (c) for all i, $1 \le i < n$ there is an rwtg-path between u_j and u_{j+1} with associated word in B ∪ C',

where wr-initial or wr-terminal spans are defined by interchanging r and w in the definitions of rw-initial and rw-terminal spans respectively and C' = $\{\vec{t}^{*}\vec{w} \cup \overleftarrow{r}\overleftarrow{t}^{*} \cup \vec{t}^{*}\vec{w}\overleftarrow{r}\overleftarrow{t}^{*}\}$. Of course, *can•tell•f*(p,q,G) can be similarly defined.

Acknowledgements: This paper benefitted from hours of discussion with T. Budd, R. W. Fabry, A. K. Jones and W. L. Ruzzo. We must thank W. L. Ruzzo and the referees for important technical suggestions, and Michele Boucher for her patience and expert typing of the document.

8. *References*

[1] G. S. Graham and P. J. Denning. Protection - principles and practices. *Proceedings of SJCC*, pp.417-429, 1972.

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

[3] *The Concise Oxford English Dictionary.* Oxford University Press, Sixth Edition, 1976.

----

*The names are not at all suggestive, however.

[4] Butler W. Lampson.
A note on the confinement problem.
CACM 16(10):613-615.

[5] M. A. Harrison, W. L. Ruzzo and J. D. Ullman.
On Protection in Operating Systems.
CACM 19(8):461-471 (August, 1976).

[6] A. K. Jones, R. J. Lipton and L. Snyder.
A linear time algorithm for deciding security.
Proceedings of the 17th Annual Symposium on Foundations of Computer Science, 1976.

[7] L. Snyder.
Formal Models of Capability-Based Protection Systems.
Yale Department of Computer Science Technical Report, #151, 1978.

[8] R. W. Fabry, private communication.

[9] W. L. Ruzzo, private communication.

[10] Matt Bishop and Lawrence Snyder.
The Transfer of Information and Authority in a Protection System.
Yale Department of Computer Science Technical Report, #166, 1979.

[11] Dorothy E. Denning.
A Lattice Model of Secure Information Flow.
CACM 19(5):236-243 (May, 1976).

[12] Anita K. Jones and Richard J. Lipton.
The enforcement of security policies for computation.
JCSS 17(1):35-55 (January, 1978).