Proceedings of Sixth ACM Symposium on Operating Systems Principles (November 1977) 133-139.

INFORMATION TRANSMISSION IN COMPUTATIONAL SYSTEMS

Ellis Cohen
University of Newcastle upon Tyne

This paper presents Strong Dependency, a formalism based on an information theoretic approach to information transmission in computational systems. Using the formalism, we show how the imposition of initial constraints reduces variety in a system, eliminating undesirable information paths. In this way, protection problems, such as the Confinement Problem, may be solved. A variety of inductive techniques are developed useful for proving that such solutions are correct.

Section 1. INTRODUCTION

Users of large computational systems require assurances that information they deem private remains private. Recent work (e.g. [Denning 76, Walters 75]) has used a notion of "information flow" in order to formally state such assurances. If, for each operation provided at some level of a system, one can determine the information flow resulting from that operation - inductive techniques can be used to determine the extent of information transmission in the system as a whole.

The information flow resulting from execution of a single operation can formally be determined from the semantics of the operation. However, except for work by Jones & Lipton [75], operational information flow has been based on the syntax of the operation instead [Denning and Denning 76, Millen 76].

This paper introduces Strong Dependency, which derives the information flow from the functional semantics of an operation. (A companion paper [Cohen 77] derives axiomatic techniques for determining the operational information flow from syntax.) Furthermore, it provides a unified theory of information transmission for both single operations and for the entire system.

Strong Dependency is based on ideas taken from classical information theory [Ashby 56]. Notably, it borrows the idea that information is transmitted from a source to a destination only when variety in the source can be conveyed to the destination. Sections 2-4 provide background definitions and formalise these ideas.

The research described in this paper was supported at Carnegie-Mellon University by the Defense Advanced Research Projects Agency (F44620-73-C-0074) where it is monitored by the Air Force Office of Scientific Research, and by the National Science Foundation under grant MCS75-07251A01.

Initial constraints on a system can decrease the variety in a system, reducing the variety conveyed and preventing transmission of information. Initial constraints are important, for current systems tend to provide fairly elaborate mechanisms [Cohen and Jefferson 75] for preventing information transmission, however, these must be initialised properly in order to have the desired effect. Sections 5 and 6 show how the Strong Dependency formalism can be used to state what must be proven - the absence of certain information transmission paths given proper initialisation.

Proving the absence of information transmission over all possible executions of a system requires an inductive proof technique. Section 7 accordingly derives Strong Dependency Induction, which is shown to be a generalisation of the information flow induction techniques mentioned above. Section 8 shows how Strong Dependency Induction might be used, by certifying a solution to the Confinement Problem [Lampson 73] in a simple system.

Information flow has generally been defined as transitive. That is, if information can flow from a to m and from m to b, then a flow from a to b is assumed as well. Section 9 illustrates two systems where transitivity fairly obviously does not hold. Moreover, a technique for Separation of Variety is derived, which can be used in conjunction with Strong Dependency Induction to prove absence of information transmission in non-transitive cases.

Section 2. COMPUTATIONAL SYSTEMS

A Computational System is formally specified as a triple $\langle NM, \Sigma, \Delta \rangle$. We may imagine that a system is comprised of a set of objects (e.g. variables, files, processes) named by NM. The value (contents) of some object x ($x \in NM$) depends upon the state of the system σ ($\sigma \in \Sigma$). I write $\sigma.x$ to mean the value of x in state σ.

Execution of an operation alters the state. Formally we define an operation δ ($\delta \in \Delta$) as a mapping from states to states and interpret $\delta(\sigma) = \sigma*$ to mean that $\sigma*$ is the new state after δ is executed in state σ. Informally we can define operations by any convenient programming-language-like notation. For example, if σ were some state such that $\sigma.a = 11$, and δ were the operation δ: b ← a, then the state $\delta(\sigma)$ would be identical to σ except that b's value in state $\delta(\sigma)$ would be 11 ($\delta(\sigma).b = 11$), regardless of its value in state σ.

I write $\sigma 1 \underset{x}{\equiv} \sigma 2$ to mean that $\sigma 1$ and $\sigma 2$ must have the same values for all objects except x. Formally

$$\sigma 1 \underset{x}{\equiv} \sigma 2 \quad \equiv_{def} \quad (\forall y \neq x)(\sigma 1.y = \sigma 2.y)$$

In the example above, $\delta(\sigma) \underset{b}{\equiv} \sigma$.

A history is defined as a sequence of operations. If H were $\delta_1\delta_2\delta_3$ then execution of H indicates sequential execution of δ_1, δ_2 and then δ_3.

$$H(\sigma) = (\delta_1\delta_2\delta_3)(\sigma) = \delta_3(\delta_2(\delta_1(\sigma)))$$

Section 3. INFORMATION TRANSMISSION

Information transmission in computational systems can be formalised by considering the classical treatment of information theory [Ashby 56]. For example, consider execution of

δ: b ← a

with respect to an information channel having a as a source and b as a destination. Our task is to determine whether information is transmitted over that channel. Classical information theory tells us that

Information is transmitted over a channel when variety is conveyed from the source to the destination

For example, if (in a 16 bit machine) a could (with equal probability) take on any of 2**16 possible values, then there are $\log_2(2^{16}) = 16$ bits of variety in a and execution of δ conveys all of that variety to b, for after execution of δ, b may also take on 2**16 values, each one corresponding to an initial value of a.

In general, information can be transmitted from a to b over execution of H if, by suitably varying the initial value of a (exploring the variety in a), the resulting value in b after H's execution will also vary (showing that the variety is conveyed to b). Note that this formulation implies that if a is known to be a particular constant, then no information can be transmitted from a to b. If a initially contains no variety, then surely none can be conveyed to b.

Section 4. STRONG DEPENDENCY

Classical information theory is concerned with the amount of information transmitted over a channel. Strong Dependency is only concerned with whether any information can be transmitted at all. I write $a \triangleright^H b$, b Strongly Depends on a over execution of H, to mean that variety in a can be conveyed to b over execution of the sequence of operations H. Formally

$$a \triangleright^H b \quad \equiv_{def} \quad (\exists \sigma 1 \sigma 2) \; (\; \sigma 1 \underset{a}{\equiv} \sigma 2 \; \wedge \; H(\sigma 1).b \neq H(\sigma 2).b \;)$$

$\sigma 1 \underset{a}{\equiv} \sigma 2$ represents two suitably chosen states that differ at a alone - i.e. exhibiting variety in a. That variety can be conveyed to b, if after execution of H, the value of b differs in the two states.

Figure 1 presents two examples of Strong Dependency, both showing how information can be transmitted from a to b over execution of a single operation.

Example 1: δ: b ← a

	$\sigma 1$	$\sigma 2$	$\delta(\sigma 1)$	$\delta(\sigma 2)$
a	10	20	10	20
b	40	40	10	20

Example 2: δ: if a then b ← x

	$\sigma 1$	$\sigma 2$	$\delta(\sigma 1)$	$\delta(\sigma 2)$
a	true	false	true	false
b	40	40	60	40
x	60	60	60	60

$\sigma 1 \underset{a}{\equiv} \sigma 2$ $\qquad$ $\delta(\sigma 1).b \neq \delta(\sigma 2).b$

$a \triangleright^{\delta} b$

Figure 1

Example two has intrigued previous researchers, for if information transmission is perceived as a "flow" of information, then when a is false, as in $\sigma 2$, execution of δ has no effect and can hardly result in a "flow" of information from a to b. Yet the fact that b's value does not change, of course indicates that a must be false, representing new information gained about a. Jones and Lipton [75] and Denning [76] discuss these situations using the terms "negative inference" and "implicit flow" respectively. These cases need not be treated specially. Using Strong Dependency, we never ask whether there is a "flow" of information when a history is executed in some particular state; rather we compare the effect of execution in different states. Analogously, in classical information theory, the information transmitted over a channel is not determined by the contents of any single message, but rather by the variety in the set of contents of all possible messages.

Section 5. TRANSMISSION AND CONSTRAINTS

This section considers the effect of constraining a system so that it is permitted to begin execution only in some subset of all possible initial

states. The approach to constraint again derives from classical information theory:

Constraint reduces the variety in a system potentially reducing the variety that may be conveyed

For example, if a system is initially constrained so that a can only take on values from $\emptyset$ to 7, then execution of

δ: b ← a

can still transmit information from a to b. This is written

$a \overset{\delta}{\underset{\varphi}{\rhd}} b$ where

$\varphi(\sigma) \equiv \emptyset \leq \sigma.a \leq 7$

However if δ' is the operation

δ': b ← a div 20 (integer division)

then, the resulting value of b is <u>always</u> $\emptyset$ when φ is satisfied initially. No variety is conveyed so imposing φ prevents information transmission from a to b and

$\neg\, a \overset{\delta'}{\underset{\varphi}{\rhd}} b$ even though $a \overset{\delta'}{\rhd} b$

The definition of Strong Dependency determined the initial variety in a by examining arbitrary pairs of states that differed at a alone. In the presence of the constraint φ, we need only consider states that satisfy φ. Formally

$$\sigma 1 \overset{\varphi}{\underset{x}{\equiv}} \sigma 2 \equiv_{def} \varphi(\sigma 1) \wedge \sigma 1 \underset{x}{\equiv} \sigma 2 \wedge \varphi(\sigma 2)$$

$$a \overset{H}{\underset{\varphi}{\rhd}} b \equiv_{def} (\exists \sigma 1 \sigma 2)\ (\ \sigma 1 \overset{\varphi}{\underset{a}{\equiv}} \sigma 2 \wedge H(\sigma 1).b \neq H(\sigma 2).b\ )$$

The examples above showed how information transmission could be prevented by reducing the variety in the source a. A more traditional sort of contraint has been discussed previously by Millen [76], who notes that execution of

δ: <u>if</u> q <u>then</u> b ← a

causes no transmission of information from a to b if q is constrained to be false. Formally, it can be shown that

$\neg\, a \overset{\delta}{\underset{\varphi}{\rhd}} b$ where

$\varphi(\sigma) \equiv \neg\, \sigma.q$

I've implied that $\neg\, a \overset{H}{\underset{\varphi}{\rhd}} b$ guarantees that information cannot be transmitted from a to b over transmission of H given <u>any</u> φ. In fact, this guarantee may not hold for constraints that are not <u>autonomous</u>. A formal definition of autonomy and additional details can be found in [Cohen 76]. Informally, φ is autonomous if the constraints on the values of each object are independent.

Examples of autonomous constraints are:

$\varphi(\sigma) \equiv \sigma.a \neq 7 \wedge \sigma.b = 12$
$\varphi(\sigma) \equiv (\forall x)(\ \sigma.x \leq 18\ )$

Non autonomous constraints are:

$\varphi(\sigma) \equiv \sigma.a \leq \sigma.b$
$\varphi(\sigma) \equiv (\forall a,b)(b \in \sigma.a.refs \supset \sigma.a.level = \sigma.b.level\ )$

where refs and level are distinct subcomponents of state σ. This last constraint can be read as: If b is in the set of objects a can reference in state σ, then the (dynamic security) level of a and b are the same. The dynamic nature of levels requires a non-autonomous constraint. If security levels were statically associated with objects, level(a) indicating a's level, the constraint <u>would</u> be autonomous.

$\varphi(\sigma) \equiv (\forall a,b)(\ b \in \sigma.a.refs \supset level(a) = level(b)\ )$

The value of a in state σ is dependent upon the properties of other objects (i.e. level(b)), but independent of their value. The remaining examples in this paper will all use autonomous constraints. (See the Appendix for effects of non-autonomy.)

Section 6. INFORMATION PROBLEMS AND SOLUTIONS

This section discusses how the Strong Dependency formalism can be used to state two well known problems - the Security Level Problem [Denning 76] and the Confinement Problem [Lampson 73] in a manner that is independent of any particular computational system.

The last section discussed the operation

δ: <u>if</u> q <u>then</u> b ← a

and showed that although $a \overset{\delta}{\rhd} b$, imposition of the constraint

$\varphi(\sigma) \equiv \neg\, \sigma.q$

prevented transmission of information from a to b. We can think of φ as <u>solving</u> the information problem

$\neg\, a \overset{\delta}{\underset{\varphi}{\rhd}} b$

That is - guarantee that no information may be transmitted from a to b. A solution to the problem is represented by a constraint φ whose imposition prevents the specified information transmission. φ can be thought of as characterising what are often referred to as "secure states".

The Confinement problem is concerned with guaranteeing the privacy of information held by individual users. The problem is typically described in terms of the difficult situation where a user engages a service. The private information must be passed to that service and so seems to be temporarily out of direct control of

the user. A number of solutions to this problem have been proposed [Rotenberg 73, Cohen & Jefferson 75, Lampson 73]. This paper will concentrate on a more general formulation of the problem instead.

Objects (including executors) can be divided into three groups. Those characterised as "Private" corresponding to objects directly under the control of the owners of the information. "Spy" characterises the objects to which that information must not be transmitted. The remainder of the objects can temporarily hold private information as long as it is not subsequently transmitted to a spy.

The confinement problem can then be specified as

$$(\forall x,y)\ (\ x \not\rhd_{\varphi} y \ \supset .\ \ Private(x) \supset \neg Spy(y)\ )$$

where $x \not\rhd_{\varphi} y \ \equiv_{def}\ (\exists H)\ (\ x \not\rhd^{H}_{\varphi} y\ )$

That is, if execution of any history transmits information from a private object to some other object, then that other object must not be a spy.

Of course, each user of a system may define confinement differently, in terms of different sets of "private" and "spy" objects; each one defining a different instance of the Confinement problem. Each instance of the problem determines a different set of solutions φ. Thus a system itself does not solve the Confinement problem. Rather a system may provide mechanisms that permit a user to produce secure states and which then guarantees that subsequent states remain secure.

The Security level problem is usually described as having less of an individualistic flavour. Each object x is initially assigned a security level, Level(x) determined on a system wide basis. No information is to be transmitted from an object at one level to an object at a lower level. Formally

$$(\forall x,y)\ (\ x \not\rhd_{\varphi} y \ \supset\ Level(x) \leq Level(y)\ )$$

Section 7. STRONG DEPENDENCY INDUCTION

Proving the correctness of solutions to the Confinement problem or the Security level problem requires an analysis of information paths over all possible histories. Strong Dependency Induction makes that task practical. Intuitively, it is based on the notion that if information is transmitted from a to b over execution of HH', then there must be some intermediate object m (perhaps the same as a or b in degenerate cases) such that execution of H transmits information from a to m and execution of H' transmits information from m to b. Formally it can be proven that the following property holds.

PROPERTY 1 (Flow Property)

$$a \not\rhd^{HH'} b \ \supset\ (\exists m)\ (\ a \not\rhd^{H} m\ \wedge\ m \not\rhd^{H'} b\ )$$

When an initial constraint φ is imposed on the system, the theorem continues to hold as long as φ is both invariant (that is $(\forall\sigma,\delta)(\ \varphi(\sigma) \supset \varphi(\delta(\sigma))\ )$ and autonomous (other generalisations may be found in [Cohen 76]).

PROPERTY 2 (Flow with Constraint)

If φ is invariant and autonomous then

$$a \not\rhd^{HH'}_{\varphi} b \ \supset\ (\exists m)\ (\ a \not\rhd^{H}_{\varphi} m\ \wedge\ m \not\rhd^{H'}_{\varphi} b\ )$$

The following property follows directly.

PROPERTY 3 (End point Isolation)

If φ is invariant and autonomous then:

To show $\neg\ a \not\rhd_{\varphi} b$ prove

$(\forall m\neq b,\delta)\ (\ \neg\ m \not\rhd^{\delta}_{\varphi} b\ )$ or

$(\forall m\neq a,\delta)\ (\ \neg\ a \not\rhd^{\delta}_{\varphi} m\ )$

That is, if no single operation can transmit information to b from any other object or if no single operation can transmit information from a to any other object, then information cannot be transmitted from a to b over any history.

More generally, we may want to show that $x \not\rhd_{\varphi} y$ can be permitted, but only for certain classes of $\langle x,y\rangle$ pairs, for example when Level(x) $\leq$ Level(y). That relationship is both reflexive and transitive, and in general

PROPERTY 4 (Lattice Induction)

If φ is invariant and autonomous and q is reflexive and transitive then

To show $(\forall x,y)\ (\ x \not\rhd_{\varphi} y \ \supset\ q(x,y)\ )$

Prove $(\forall x,y,\delta)\ (\ x \not\rhd^{\delta}_{\varphi} y \ \supset\ q(x,y)\ )$

Thus for the Security level problem, it is only necessary to show that if x's security level is greater than y's, no single operation transmits information from x to y.

Reflexive, transitive relations arise naturally in the definition of information problems. As a result, a number of researchers [Denning 75, Walters 75, Jones & Lipton 76] have suggested an information flow relation that is itself reflexive and transitive. Such a relation is analogous to Strong Dependency having the property that

$$a \not\rhd^{H} m\ \wedge\ m \not\rhd^{H'} b\ \supset .\ \ a \not\rhd^{HH'} b$$

as noted by Denning [75], this property does not always hold, and can indicate that information may be transmitted even when no such transmission is possible. In section 9 we will explore just such a situation and develop a technique suitable for handling it correctly.

Section 8. AN EXAMPLE OF STRONG DEPENDENCY INDUCTION

This section illustrates Strong Dependency Induction by proving the correctness of a solution to the Confinement Problem in a simple system. Imagine a system where each object contains two disjoint subcomponents, one holding "data" and one holding a pointer to another object: The system has two sets of operations:

$\delta 1(y,x)$: if y.ptr = x then y.data ← x.data
$\delta 2(y,x)$: if y.ptr = x then y.ptr ← x.ptr

If y points to x, then execution of $\delta 1(y,x)$ will copy data from x to y. If y points to x and x points to w, then after execution of $\delta 2(y,x)$, y will point to w as illustrated in figure 2.

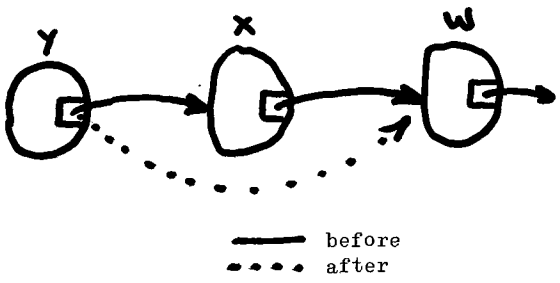

Figure 2

If it is assumed that any object that is not "private" is a "spy", the Confinement problem can be stated as:

$$(\forall x,y)\ (\ x \triangleright_{\varphi} y \supset.\ \ Private(x) \supset Private(y)\ )$$

I'll show that as long as a spy doesn't point to a private object, confinement is enforced. Formally the initial constraint φ can be chosen to be

$$\varphi(\sigma) \equiv (\forall y)\ (\ Private(\sigma.y.ptr) \supset Private(y)\ )$$

That is, if in any state σ, an object y points to a private object, then y must be private

φ is autonomous and can easily be shown to be invariant (the only interesting case is $\delta 2(y,x)$).

Next, pick

$$q(x,y) \equiv Private(x) \supset Private(y)$$

noting that q is both reflexive and transitive. It can be easily shown that

$$(\forall x,y,\delta)\ (\ x \triangleright^{\delta}_{\varphi} y \supset q(x,y)\ )$$

By Lattice Induction, confinement is enforced.

Section 9. SEPARATION OF VARIETY

Strong Dependency Induction may fail to be useful in proving the absence of information transmission if the Strong Dependency relation is not transitive. This section explores two non-transitive examples and derives a new technique, Separation of Variety, which extends the applicability of Strong Dependency Induction.

Consider the system

$\delta 1$: if q then m ← a
$\delta 2$: if ¬ q then b ← m

Information cannot be transmitted from a to b in this system. If q is false, no information can be obtained from a; if q is true, no information can be stored in b. Formally, it can be shown that

$\neg\ a \triangleright^{\delta 1\delta 2} b$ even though

$a \triangleright^{\delta 1} m$ and $m \triangleright^{\delta 2} b$

Thus an attempt to prove $\neg\ a \triangleright b$ directly by End point Isolation is doomed to failure, for property 3 requires that

$(\forall \delta, x \neq a)\ (\ \neg\ a \triangleright^{\delta 1} x\ )$ [yet $a \triangleright^{\delta 1} m$] or

$(\forall \delta, x \neq b)\ (\ \neg\ x \triangleright^{\delta 2} b\ )$ [yet $m \triangleright^{\delta 2} b$]

The informal explanation above which argued that information could not be transmitted from a to b proceeded by distinguishing two cases - q false and q true. This can be formalised in the following way. Let

$\varphi 1(\sigma) \equiv \neg\ \sigma.q$
$\varphi 2(\sigma) \equiv \sigma.q$

Though $a \triangleright^{\delta 1} m$, $\neg\ a \triangleright^{\delta 1}_{\varphi 1} m$ and

$(\forall \delta, x \neq b)\ (\ \neg\ a \triangleright^{\delta}_{\varphi 1} x\ )$

so by property 3, $\neg\ a \triangleright_{\varphi 1} b$.

Similarly $\neg\ m \triangleright^{\delta 2}_{\varphi 2} b$ which is the main result needed to demonstrate $\neg\ a \triangleright_{\varphi 2} b$.

Since $\varphi 1$ and $\varphi 2$ cover all cases, $\neg\ a \triangleright b$.

Not every separation into cases is legitimate. Consider the system

δ: if a then b ← 0 else b ← 1

Pick

$\varphi 1(\sigma) \equiv \neg\ \sigma.a$
$\varphi 2(\sigma) \equiv \sigma.a$

Then $\neg\ a \triangleright_{\varphi 1} b$ and $\neg\ a \triangleright_{\varphi 2} b$ yet information certainly can be transmitted from a to b. The trouble is that $\varphi 1$ and $\varphi 2$ are dependent upon the value of a - the very object that is in question as an information source. Separation of variety is legitimate only when each of the separating constraints is independent of the information source in question. Define

φ is a-independent $\equiv_{def}$

$$(\forall \sigma 1, \sigma 2)\ (\ \sigma 1 \underset{a}{=} \sigma 2 \supset.\ \ \varphi(\sigma 1) = \varphi(\sigma 2)\ )$$

If there is any set of constraints $\{\varphi i\}$, each a-independent, not necessarily disjoint, but such that any state is satisfied by at least one φi - then - if no information is transmitted from a to b given each of the φi's in turn, then no information can be transmitted from a to b at all.

PROPERTY 5 (Separation of variety)

If $(\forall \sigma \exists i)\ (\ \varphi i(\sigma)\ )$ [to guarantee that φi covers all cases]

and $(\forall i)\ (\ \varphi i$ is a-independent $)$

then $(\forall i)\ (\ \neg\ a\ \|\rhd^{H}_{\varphi i}\ b\ )\ \supset\ \neg\ a\ \|\rhd^{H}\ b$

or more generally

PROPERTY 6 (Separation with Constraint)

If $(\forall \sigma \exists i)\ (\ \varphi i(\sigma)\ )$ [to guarantee that φi covers all cases]

and $(\forall i)\ (\ \varphi i$ is a-independent $)$

then $(\forall i)\ (\ \neg\ a\ \|\rhd^{H}_{\varphi \wedge \varphi i}\ b\ )\ \supset\ \neg\ a\ \|\rhd^{H}_{\varphi}\ b$

Another use of separation of variety arises in showing that no information can be transmitted from a to b in the system

$\delta 1$: m ← m + k*a
$\delta 2$: b ← m mod k

Pick $\varphi i(\sigma)\ \equiv\ (\ \sigma.m \bmod k\ ) = i$

First $(\forall i)\ (\ \neg\ m\ \|\rhd^{\delta 2}_{\varphi i}\ b\ )$ though $m\ \|\rhd^{\delta 2} b$

more generally $(\forall i)\ (\forall x \neq b, \delta)\ (\ \neg\ x\ \|\rhd^{\delta}_{\varphi i}\ b\ )$

Since each φi can be shown to be invariant, End point Isolation can be used to show that

$(\forall i)\ (\ \neg\ a\ \|\rhd_{\varphi i}\ b\ )$

Finally, by Separation of variety $\neg\ a\|\rhd\ b$.

This example is especially interesting, for showing $\neg\ m\ \|\rhd^{\delta 2}_{\varphi i}\ b$ takes advantage of the fact that information transmission may be eliminated by sufficiently reducing the variety in the source (section 5).

Section 10. CONCLUSION

This paper has presented Strong Dependency, a formalism for describing information transmission in computational systems and for proving that information is not transmitted over certain paths. Attention was restricted to systems constrained by a class of predicates which are both autonomous and invariant and the paper considered whether any information at all might be transmitted over an information channel, without attempting to measure the channel capacity. A discussion of the more general situation may be found in [Cohen 76].

In analysing transmission from a to b, the paper considered the effect that a change in a might have on b over any arbitrary sequence of operations. If we are concerned with information transmission as the result of execution of a sequential program, we instead need consider the effect on b over executions of the program. The resulting theory is discussed in [Cohen 77].

ACKNOWLEDGEMENTS

It's a pleasure to thank Dorothy Denning for her comments and to thank the whole host of members of the CMU community for many discussions of this material.

REFERENCES

W. Ashby. "An Introduction to Cybernetics", 1956.

E. Cohen. "Strong Dependency: A Formalism for Describing Information Transmission in Computational Systems", Carnegie-Mellon Univ., Comp. Sci. Tech. Report, August 1976.

E. Cohen. "Information Transmission in Sequential Programs", Submitted for publication 1977.

E. Cohen, D. Jefferson. "Protection in the HYDRA Operating System", Proc. 5th Symp. on Operating Systems Principles, November 1975.

D. Denning, "Secure Information Flow in Computer Systems", Ph.D. Thesis, Comp. Sci. Dept., Purdue Univ., May 1975.

D. Denning, "A Lattice Model of Secure Information Flow", CACM v. 19, 5, (May 1976).

D. Denning, P. Denning, "Certification of Programs for Secure Information Flow", Purdue Univ., CSD-TR 181, March 1976.

A. Jones, R. Lipton. "The Enforcement of Security Policies for Computations", Proc. 5th Symp. on Operating Systems Principles, November 1975.

B. Lampson. "A Note on the Confinement Problem", CACM v. 16, 10 (October 1973).

J. Millen. "Security Kernal Validation in Practice", CACM v. 19, 5 (May 1976).

L. Rotenberg. "Making Computers Keep Secrets", Ph.D. Thesis MIT, MAC-TR-116, September 1973.

K. Walters, et al. "Structural Specification of a Security Kernel", 1975 Intl. Conf. on Reliable Software, April 1975.

APPENDIX: NON-AUTONOMOUS CONSTRAINTS

Strong Dependency is not a wholly adequate formalism for determining information transmission given non-autonomous constraints. Consider the system

δ: $b \leftarrow a1$

constrained by

$\varphi(\sigma) \equiv \sigma.a1 = \sigma.a2$

It can be shown that $\neg\ a1 \not\rhd_{\varphi} b$. However, information is certainly transmitted from a1 to b. An observer of b, knowing that φ holds, still gains information about a1 (its value!) from execution of δ.

The definition of Strong Dependency considers variety in a1 alone, but φ eliminates this variety by forcing a1 to remain the same as a2. The effect is similar to constraining a1 to be a constant. φ essentially spreads a1's variety to a2. To deal with the spread, Strong Dependency can be extended to allow a <u>set</u> of objects as an information source, and autonomy can be considered relative to that set alone [Cohen 76]. In the example above, φ is said to be $\{a1,a2\}$-autonomous and $\{a1,a2\} \not\rhd_{\varphi} b$.

Formal methods for showing that b depends on a1 - without needing to explicitly be concerned about a2 - can be derived using statistical inference methods. That is, however, beyond the scope of this paper.