STRONG DEPENDENCY: A Formalism for Describing Information Transmission in Computational Systems

Ellis S. Cohen

Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
August 1976

ABSTRACT

This paper presents an information theoretic approach to information transmission in computational systems. We formalize the effect of constraint on information paths and develop a number of inductive techniques for proving the absence of information transmission. Finally, we show how ordinary inductive assertions can be used in conjunction with the theory to analyze information paths in sequential programs.

This work was supported 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.

TABLE OF CONTENTS

INDEX OF DEFINITIONS

Strong Dependency

53 6-2 $\emptyset$ is an inductive cover for φ

Chapter 1 - Introduction [inftro:]

----- Section 1.1 --- Motivation []

This paper introduces Strong Dependency, a formal theory of information transmission in computational systems. We develop a number of proof techniques and show how they can be used in solving information problems.

The need to study information transmission arose from our work on the Confinement Problem [Lampson 73]. Imagine that some user of a service has to tell the service personnel private information. The user wants to guarantee that the information is kept private. That is, no information is to be transmitted from the program executing the "service" to anyone but the "user" (or perhaps other "users" designated by her).

We believed that the protection mechanism developed for the Hydra Operating System [Wulf 74, Cohen & Jefferson 75] allowed the construction of an elegant solution to the Confinement Problem. However, in order to prove that a solution to the Confinement Problem was indeed correct, we needed to develop a formal theory of information transmission in computational systems. This paper introduces the basics of such a formalism, and presents a number of examples to illustrate its use.

----- Section 1.2 --- Computational Systems [compsys:]

We have defined a computational system [Cohen 76] as a pair, $<\Sigma,\Delta>$, where $\sigma \in \Sigma$ is a state of the system, and $\delta \in \Delta$ is an operation.

Each state is wholly comprised of a set of object, each having a fixed unique name. If α is the name of some object, we write $\sigma.\alpha$ to mean the value of α in state σ. Formally, a state is a vector of objects

$$\sigma \equiv <\sigma.n1, \sigma.n2, \ldots>$$

where $<n1,n2,..>$ are the list of object names in lexicographic order. If A is a set of object names, we write $\sigma.A$ to mean

$$\sigma.A \equiv \langle \sigma.\alpha 1, \sigma.\alpha 2, \ldots \rangle$$

where $\langle \alpha 1, \alpha 2, \ldots \rangle$ are the list of names in A in lexicographic order. This definition permits us to write

$$\sigma 1.A = \sigma 2.A \quad \text{for} \quad (\forall \alpha \in A)(\ \sigma 1.\alpha = \sigma 2.\alpha\ )$$

We define

>> Def 1-1) $\sigma 1 \underset{A}{=} \sigma 2$

$$\sigma 1 \underset{A}{=} \sigma 2 \equiv_{def} (\forall \alpha \notin A)(\ \sigma 1.\alpha = \sigma 2.\alpha\ )$$

That is, if $\sigma 1 \underset{A}{=} \sigma 2$, then states $\sigma 1$ and $\sigma 2$ may differ only in the values of the objects named by A. For the special case, where $\sigma 1$ and $\sigma 2$ may differ only in the value of a single object α, we define

>> Def 1-2) $\sigma 1 \underset{\alpha}{=} \sigma 2$

$$\sigma 1 \underset{\alpha}{=} \sigma 2 \equiv_{def} (\forall \alpha' \neq \alpha)(\ \sigma 1.\alpha' = \sigma 2.\alpha'\ )$$

Objects may themselves have some internal structure (including pointers to other objects). However, such details are part of an interpretation and not part of our abstract model. As an example though, we might write $\sigma.x.k$ to mean the value of the k'th component of object x in state σ.

We formally define an operation δ as a function from states to states. Semantically, we interpret $\delta(\sigma) = \sigma'$ to mean that execution of δ in state σ may alter some objects in the state to produce a new state σ'. We find it useful to describe operations in terms of an informal programming-like language. For example, if σ' were just like σ, except that $\sigma'.\beta = \sigma.\alpha$, we could write

$$\delta: \quad \beta \leftarrow \alpha$$

An history is a sequence of operations (e.g. $\delta 1 \delta 2 \delta 3$). When a history is applied to a state, the operations in the history are applied sequentially from left to right. Formally we define

>> Def 1-3] $H(\sigma)$ (recursively defined)

$\lambda(\sigma) \leftarrow\!= \sigma$ (λ is the null history - no operations)

$(H\delta)(\sigma) \leftarrow\!= \delta(H(\sigma))$

We write both HH' as well as $H \& H'$ to mean the concatenation of the sequences H and H' (note "&" is not commutative).

If a system is started in state σ and some arbitrary sequence of operations H is executed, then the system exhibits some behavior which can be completely described by the pair $\langle\sigma, H\rangle$. We call a pair $\langle\sigma, H\rangle$ a behavior or a computation.

----- Section 1.3 --- Access Matrix Systems [mtrx:]

Protection in operating systems is often modelled using a matrix of protection rights [Lampson 71]. Briefly, we describe such a model as follows: Before any operation permits an object to be accessed in some way, the matrix is checked to determine whether the executor of the operation has the appropriate right for that access. For example, if execution of some operation would permit Cohen to write into the Salary file, the operation would first check that Cohen has the right to write the Salary file, notationally

$$w \in \langle \text{Cohen}, \text{Salary} \rangle(\sigma)$$

That is, are w (write) rights to be found in the <Cohen,Salary> entry of the protection matrix in state σ?

In this paper, we will occasionally refer to a simple system having three rights, s (subject), r (read) and w (write) interpreted as

$s \in \langle x, x \rangle(\sigma)$	allows x to execute operations in state σ
$r \in \langle x, \alpha \rangle(\sigma)$	allows x to read file α in state σ
$w \in \langle x, \beta \rangle(\sigma)$	allows x to write file β in state σ

A operation copy(user,fnew,fold) that allows "user" to copy the contents of fold to fnew might be defined as

copy(user,fnew,fold): if s ∈ <user,user>
∧ r ∈ <user,fold>
∧ w ∈ <user,fnew>
then fnew ← fold

That is, "user" must be able to execute (be a subject), read fold, and write fnew.

----- Section 1.4 --- Behavioral Problems [behprob:]

We showed in [Cohen 76] that many problems ordinarily considered to be protection problems can be formally characterized as constraints on the behavior of the system. Consider the problem: Cohen is not to be able to write the Salary file. This problem characterizes those behaviors of the system as "acceptable", which when executed, do not execute any operations that have the effect of causing a write access of the Salary file by Cohen. We can write $\Psi_{problem}$ to characterize these acceptable behaviors, where

$$\Psi_{problem}(\sigma,\lambda) \iff tt$$

$$\Psi_{problem}(\sigma,H\delta) \iff \Psi_{problem}(\sigma,H) \wedge \neg Wacc(Cohen,Salary)(H(\sigma),\delta)$$

where $Wacc(x,\beta)(\sigma,\delta)$ is defined so that when operation δ is executed in state σ, a write access is made by x to β

We say that such a problem is an enforcement problem and that it may be solved by appropriately constraining the initial state of the system. These initial states are "secure" in that, no matter what sequence of operations are subsequently executed, the behavior executed (determined by <initial state,sequence of operations>) is guaranteed to be acceptable. Formally, if φ_{solve} characterizes these secure initial states, then we say that φ_{solve} enforces $\Psi_{problem}$ where

>> Def 1-4] Φ_{solve} enforces $\Psi_{problem}$ iff

$$(\forall \sigma, H)\ (\ \Phi_{solve}(\sigma) \supset \Psi_{problem}(\sigma, H)\ )$$

Information problems are concerned with preventing the transmission of information and are fundamentally different than the enforcement problems described above. For example, a solution to the Salary file problem defined above does not necessarily solve the problem: No information is to be transmitted from Cohen to the Salary file. Cohen may be able to place information in some other file, where a confederate may write it to the Salary file.

It is tempting to try to describe the information problem as an enforcement problem as well. Suppose we write

$$\alpha\text{-}(\sigma\text{:}H)\text{->}\beta$$

to mean that information flows from α to β over execution of behavior $<\sigma, H>$. Then we can describe the information transmission problem formally as

$$\Psi_{problem}(\sigma, H) \equiv \neg\ \text{Cohen-}(\sigma\text{:}H)\text{->Salary}$$

However, we need first to be able to define the meaning of $\alpha\text{-}(\sigma\text{:}H)\text{->}\beta$. Such a definition is difficult for the following reason. Suppose that some operation δ caused β to be written into only if some property p held true of α, and that p did not hold true of α in state σ. We might naively conclude that $\neg\alpha\text{-}(\sigma\text{:}H)\text{->}\beta$. However, an observer of β may note that β is not written into and may therefore conclude that property p does not hold true of α. Even though β has not actually been written into, information about α (the fact that p holds true of α) is nonetheless transmitted to β in state σ.

[Jones & Lipton 75] have described such situations by the term "negative inference". [Denning 75] has termed such information transmission "implicit flow" as distinguished from the case where β is explicitly written into.

There are a number of solutions to this dilemna. One might define $\alpha\text{-}(\sigma\text{:}H)\text{->}\beta$ in such a way that implicit flow is taken into consideration. In

[Cohen 76], we argue that such an approach is inappropriate - that the determination of acceptable behaviors (and thus the information transmitted) is actually determined in part by the constraints (i.e. φ_{solve}) placed on the system.

----- Section 1.5 --- Models of Information Transmission [modinf:]

[Denning 75] and [Case 74] have gotten rid of the problem of implicit flows by disregarding the state in which an operation is executed. Information is considered to flow from α to β over execution of δ (which we can write as $\alpha\text{-}(\delta)\text{->}\beta$) so long as there exists some state in which execution of δ explicitly transmits information from α to β.

Information flow of a sequence of operations is defined by assuming that information flow is transitive. That is, information flow is defined recursively as

$$\alpha\text{-}(\lambda)\text{->}\beta \iff (\alpha = \beta) \qquad (\text{"}\lambda\text{" is the null history})$$

$$\alpha\text{-}(H\delta)\text{->}\beta \iff (\exists m)(\alpha\text{-}(H)\text{->}m \wedge m\text{-}(\delta)\text{->}\beta)$$

where m may be the same as α or β [e.g. $m\text{-}(\delta)\text{->}m$ as long as the execution of δ does not completely overwrite m]. It must be noted that in [Case 74], no definition is given for $\alpha\text{-}(\delta)\text{->}\beta$; it is left to the reader's intuition. Denning, in [Denning 75], shows how information flow may be defined for a particular programming language, but again, (though the definition of $\alpha\text{-}(\delta)\text{->}\beta$ must conform to certain theoretical considerations), it does not not derive from a theoretical formulation of the meaning of information transmission. In this paper, we will show how such a definition may be derived from the semantics of a given operation, though we will use the notation

$$\alpha \; \mathbb{D}^{\delta} \; \beta.$$

The assumption of transitivity in defining information flow over sequences of operations turns out to be a dangerous one. Consider the sequence of operations $\delta 1 \delta 2$, where

δ1: if q then m ← α
δ2: if ¬q then β ← m

α-(δ1)->m and m-(δ2)->β. By transitivity, α-(δ1δ2)->b, though it is clear that no information can in fact be transmitted from α to β. We shall introduce a technique we call separation of variety to handle such non-transitive situations.

In effect, an execution of δ2 that transmits information from m to β must occur in an environment in which q is false, but in such an environment, no information could have been transmitted from α to m by δ1. We may formally characterize an environment by a constraint φ, that cooresponds to an assertion about the state in which an operation is to be executed. We suggested above that the constraint itself must be used in determining what information transmission takes place. [Millen 76] has explored such an approach and has shown how certain information paths may be ignored in the face of appropriate constraints. We will also be studying information transmission in the presence of constraints, formally validating the approach and determining (in discussing non-autonomous constraints) its limits (which determines the limits of Millen's approach as well).

The work of both Denning and Millen is directed primarily towards analysis of information paths in sequential programs. We will be concerned more generally with analysis of information paths and the solution of information problems (determining how certain information paths may be eliminated) in arbitrary computational systems, considering sequential programs as a special case.

----- Section 1.6 --- Strong Dependency [istrdep:]

In this paper, we introduce the Strong Dependency formalism as a means of characterizing information transmission in computational systems. Strong Dependency is not an information flow model. Instead, it is based on a cybernetic or information theoretic approach to information transmission.

We imagine that each object in system may take on a set of values; this is known as the variety of the object. Information can be transmitted from

one object to another if the variety of the first object can be conveyed to the other object.

> [Our formal definition of Strong Dependency is similar to a formalism introduced by [Jones & Lipton 75], though their approach was not an information theoretic one. They argue that no information is transmitted from α to β in some system if that system can be transformed into another system with the following property: the values taken on by β are the same in both systems, but the transformed system does not access α. In effect the Strong Dependency formalism compares the original system not with a transformed system, but with a system just like the original one except that α takes on an arbitrarily different initial value.]

Next we show, that by placing an initial constraint on a system, we may reduce the variety in an object. If the variety is sufficiently reduced, no variety may be conveyed, and no information can be transmitted.

We find that Strong Dependency only corresponds to information transmission in systems constrained by certain classes of constraints. Progress on theories that correspond to information transmission in systems with arbitrary constraints is discussed in section 7.2.

----- Section 1.7 --- Plan of the Paper [plan:]

In chapter 2, we discuss the details of the Strong Dependency formalism. In chapter 3 we show how the Strong Dependency formalism can be used to define information problems including the Confinement Problem. We define a solution to an information problem as a constraint that eliminates information transmission as required by the description of the problem. We also present a measure based on Strong Dependency for comparing and evaluating solutions.

In chapter 4 we introduce Strong Dependency Induction, a technique for showing that certain classes of solutions (constraints) solve information problems. We also formally develop Separation of Variety, a technique for handling non-transitivity in information transmission. In chapters 5 and 6

we extend the class of constraints that can be used with Strong Dependency Induction. In chapter 6, we also show how Strong Dependency Induction can be used to explore information transmission in the execution of sequential programs. Chapter 7 discusses other work in progress.

----- Section 1.8 --- A Cybernetic Evaluation []

Cybernetics first [Ashby 56] formalized the idea that information transmission has nothing to do with the content of the messages transmitted, but depends only upon the way "variety" is conveyed. Information theory represents one direction taken based on that approach; it analyzes the amount of variety conveyed from one object to another in small noisy systems. We will pursue a different course. We consider whether any variety is conveyed at all from one object to another in large noiseless systems.

Our task is compounded by the fact that there is neither a single source nor a single receiver. Each object in the system may potentially receive information from, or send information to any other object in the system. Work in progress (see section 7.4) is directed toward extending classical information theory in the directions suggested by this paper.

An information theoretic approach is probably useful; one may not in general be able to completely prevent information transmission in a system designed to be kind to users. In particular, consider a user who leaks information by execution of some peculiar sequence disk operations [Lampson 73]. One might simply be satisfied to introduce enough noise to guarantee that the bandwidth from the user to the disk is sufficiently low.

For the purposes of this paper, we will generally ignore these quantitative issues; we only explore whether any information at all can be transmitted from one object to another.

Chapter 2 - Strong Dependency & Information Transmission [strinf:]

----- Section 2.1 --- Introduction []

In this chapter, we introduce the Strong Dependency formalism as a means of characterizing information transmission in computational systems. We view information transmission in a cybernetic, or information theoretic sense. We imagine that each object in a system may take on a set of values; this is known as the variety of the object. Information can be transmitted from one object to another if the variety of one object can be conveyed to another object.

We argue that information can be transmitted from an object α to an object β, if for two different values of α, execution of some history might place different values in β.

Next we show, that by placing an initial constraint on a system, we may reduce the variety in an object. If the variety is sufficiently reduced, no variety may be conveyed, and no information can be transmitted. In this chapter, we only consider a class of constraints we call autonomous constraints, those which constrain the variety in an object independently of the value of other objects. Non-autonomous constraints introduce complications in our analysis that we will begin to discuss in chapter 5.

----- Section 2.2 --- Variety and Information Transmission [varinf:]

> "... At first, when one thinks of, say, a telegram arriving, one notices only the singleness of one telegram. Nevertheless, the act of 'communication' necessarily implies the existence of a set of possibilities, i.e. more than one, as the following example will show.
>
> "A prisoner is to be visited by his wife, who is not to be allowed to send him any message however simple. It is understood that they may have agreed, before his capture, on some simple code. At her visit, she asks to be allowed to send him a cup of coffee; assuming

the beverage is not forbidden, how is the warder to ensure that no coded message is to be transmitted by it? He knows that she is anxious to let her husband know whether or not a confederate has yet been caught.

"The warder will cogitate with reasonings that will go somewhat as follows: 'She might have arranged to let him know by whether the coffee goes in sweetened or not - I can stop that simply by adding lots of sugar and then telling him I have done so. She might have arranged to let him know by whether or not she sends a spoon - I can stop that by taking away any spoon and then telling him that Regulations forbid a spoon anyway. She might do it by sending tea rather than coffee - no, that's stopped because, as they know, the canteen will only supply coffee at this time of day.' So his cogitations go on; what is noteworthy is that at each possibility he intuitively attempts to stop the communication by enforcing a reduction of the possibilities to one - always sweetened, never a spoon, coffee only, and so on. As soon as the possibilities shrink to one, so soon is communication blocked, and the beverage robbed of its power of transmitting information. The transmission (and storage) of information is thus essentially related to the existence of a <u>set</u> of possibilities. The example may make this statement plausible; in fact it is also supported by all the work in the modern theory of communication, which has shown abundantly how essential, and how fruitful, is the concept of the set of possibilities.

"Communication thus necessarily demands a set of messages. Not only is this so, but the information carried by a particular message depends on the set it comes from. <u>The information conveyed is not an intrinsic property of the individual message</u>."

W. Ross Ashby "An Introduction to Cybernetics"

Information can be transmitted from α to β in a system if the variety, the set of values that can be taken on by α, can be conveyed to β. For example, if α and β both contain 16 bit integers (and α initially can take on each of these values with equal probability), then we might imagine that execution of

δ: $\beta \leftarrow \alpha$

would transmit 16 bits of information from α to β, and that is, in fact, correct. The set of values possible for α represent 16 bits worth of variety. All of this variety can be conveyed to β by execution of δ. After execution of δ, an observer of β can determine all (16 bits worth) of the information initially in α.

Next imagine that α is known to be a constant, say 342. No information is transmitted from α to β. There is no variety in α and so none can be transmitted to β. By executing δ, an observer of β can find out α's value. But α's value is already known! No information is transmitted at all.

In a computational system, it is not necessary that the source be constrained to be constant to prevent information transmission. Consider:

δ: if $\alpha < 10$ then $\beta \leftarrow 0$ else $\beta \leftarrow 1$

If it is known that α is always less than 10, then again no information is transmitted from α to β. Execution of δ will always set β to 0, regardless of the value of α (given that α is less than 10). If α is not so constrained, then one bit of information can be transmitted from α to β. That bit (detected by determining whether β is 0 or 1 after execution of δ) indicates whether or not α is initially less than 10. Without the constraint "α is less than 10", some information about the variety of α can be transmitted to β. With it, none is transmitted at all.

Imagine picking some state $\sigma 1$ and then some other state $\sigma 2$ that is just like $\sigma 1$ but arbitrarily varies from it in its value at α. Suppose history H is then executed and it is found that the values of β are the same regardless of whether or not H was executed in state $\sigma 1$ or $\sigma 2$. The variety in α has not been transmitted to β since the resulting value of β is the same in both cases.

Now suppose that for any pair of states, $\sigma 1$ and $\sigma 2$, that differed only at α, execution of H would result in identical values for β. Then under no circumstances could any of α's variety be conveyed to β by executing H. No information could be transmitted from α to β. Formally

>> Def 2-1] No information is transmitted from α to β by H iff

$$(\forall \sigma 1, \sigma 2)\ (\ \sigma 1 \underset{\alpha}{=} \sigma 2 \ \supset\ H(\sigma 1).\beta = H(\sigma 2).\beta\ )$$

(note from section 1.2 that $\sigma 1 \underset{\alpha}{=} \sigma 2$ means that $\sigma 1$ and $\sigma 2$ must be the same except for the value of α)

We have already seen that if the values of certain objects are known to be appropriately constrained (e.g. α is less than 10), then no information can be transmitted. We can represent this constraint by φ. For example

$$\varphi(\sigma) \ \equiv\ \sigma.\alpha < 10$$

If the variety among the states is known to be constrained by φ, the pairs of states chosen as described above need only be chosen from those that satisfy φ (e.g. - those in which α is less than 10).

>> Def 2-2] $\sigma 1 \overset{\varphi}{\underset{\alpha}{=}} \sigma 2$ iff

$$\varphi(\sigma 1) \ \wedge\ \sigma 1 \underset{\alpha}{=} \sigma 2 \ \wedge\ \varphi(\sigma 2)$$

We might then argue (not completely correctly as we shall discover in chapter 5) that, if a system is initially constrained by φ, then no information is transmitted from α to β by execution of history H as long as

$$(\forall \sigma 1, \sigma 2)\ (\ \sigma 1 \overset{\varphi}{\underset{\alpha}{=}} \sigma 2 \ \supset\ H(\sigma 1).\beta = H(\sigma 2).\beta\ )$$

----- Section 2.3 --- Strong Dependency [strdep:]

In this section, we introduce the notation of Strong Dependency.

>> Def 2-3] $\sigma 1$ and $\sigma 2$ differ only at α and differ at β after H

$$\sigma 1 \ {}_{\alpha}\Diamond^{H}_{\beta}\ \sigma 2 \ \equiv_{def}\ \sigma 1 \underset{\alpha}{=} \sigma 2 \ \wedge\ H(\sigma 1).\beta \neq H(\sigma 2).\beta$$

>> Def 2-4] β strongly depends on α after H

$$\alpha \; \mathbb{D}^{H} \; \beta \quad \equiv_{def} \quad (\exists \sigma 1, \sigma 2)\, (\, \sigma 1 \; {}_{\alpha}\Diamond^{H}_{\beta} \; \sigma 2 \,)$$

By comparing this definition with that of 2-1 above, we see that

$\alpha \; \mathbb{D}^{H} \; \beta$ iff
information can be transmitted from α to β by H

In other words, Strong Dependency is a formal definition of information transmission.

We have formalized transmission of information from one object to another. It is often useful to think of the source of information as a set of objects. For example, in

$$\delta: \quad \beta \leftarrow \alpha 1 + \alpha 2$$

we might want to say that information is transmitted from the set of objects $\{\alpha 1, \alpha 2\}$ to β. We extend the above definitions quite easily.

>> Def 2-5] $\sigma 1$ and $\sigma 2$ differ only at A and differ at β after H

$$\sigma 1 \; {}_{A}\Diamond^{H}_{\beta} \; \sigma 2 \quad \equiv_{def} \quad \sigma 1 \underset{\bar{A}}{=} \sigma 2 \;\wedge\; H(\sigma 1).\beta \neq H(\sigma 2).\beta$$

>> Def 2-6] β strongly depends on A after H

$$A \; \mathbb{D}^{H} \; \beta \quad \equiv_{def} \quad (\exists \sigma 1, \sigma 2)\, (\, \sigma 1 \; {}_{A}\Diamond^{H}_{\beta} \; \sigma 2 \,)$$

The reader may wonder whether it is possible to find that information is transmitted from some set of objects A to β and yet find that the objects in α taken sigly do not transmit information to β. That is not the case in the example above. We find both that

$\{\alpha 1, \alpha 2\} \; \mathbb{D}^{\delta} \; \beta$ as well as

$\alpha 1 \; \mathbb{D}^{\delta} \; \beta$ and $\alpha 2 \; \mathbb{D}^{\delta} \; \beta$

In general, if $A \mathbb{D}^H \beta$ and α (where $\alpha \in A$) plays any part in affecting the value of β, we will find that $\alpha \mathbb{D}^H \beta$. A formal proof of this statement requires an information theoretic argument that is part of work in progress (section 7.4). [For other comments on this example, see section 7.2.]

Using Strong Dependency alone, we can show if $A \mathbb{D}^H \beta$, at least one object in A transmits information to β. Formally

Theorem 2-1]

$$A \mathbb{D}^H \beta \supset (\exists \alpha \in A)(\alpha \mathbb{D}^H \beta)$$

Formally we say that information can be transmitted from A to β in a system if it can be transmitted from A to β over some history. We define

>> Def 2-7] β strongly depends on A

$$A \mathbb{D} \beta \equiv_{def} (\exists H)(A \mathbb{D}^H \beta)$$

----- Section 2.4 --- Strong Dependency with Initial Constraints [strphi:]

In this section, we extend the Strong Dependency formalism to cover those cases where the variety in the state space is constrained by some Φ. We extend the formalization exactly as we expanded definition 2-1 above.

>> Def 2-8] $\sigma 1$ and $\sigma 2$ are constrained by Φ and are equal except at A

$$\sigma 1 \overset{\Phi}{\underset{A}{=}} \sigma 2 \equiv_{def} \Phi(\sigma 1) \wedge \sigma 1 \underset{\bar{A}}{=} \sigma 2 \wedge \Phi(\sigma 2)$$

>> Def 2-9] $\sigma 1$ and $\sigma 2$ differ only at A and differ at β after H given Φ

$$\sigma 1 \; {}^{\Phi}_{A}\Diamond^{H}_{\beta} \; \sigma 2 \equiv_{def} \sigma 1 \overset{\Phi}{\underset{\bar{A}}{=}} \sigma 2 \wedge H(\sigma 1).\beta \neq H(\sigma 2).\beta$$

>> Def 2-10] β strongly depends on A after H given φ

$$A \mathrel{\mathbb{D}}^{H}_{\varphi} \beta \quad \equiv_{def} \quad (\exists \sigma 1, \sigma 2)(\sigma 1 \overset{\varphi}{\underset{A}{\Diamond}}{}^{H}_{\beta} \sigma 2)$$

>> Def 2-11] β strongly depends on A given φ

$$A \mathrel{\mathbb{D}}_{\varphi} \beta \quad \equiv_{def} \quad (\exists H)(A \mathrel{\mathbb{D}}^{H}_{\varphi} \beta)$$

Intuitively, no matter how a system is constrained, if β depends upon some set of objects A1 which is included in A2, then β should depend upon A2 as well, for A2 provides at least as much information. Formally

Theorem 2-2] (proof left to reader)

$$A1 \subseteq A2 \quad \supset. \quad A1 \mathrel{\mathbb{D}}^{H}_{\varphi} \beta \supset A2 \mathrel{\mathbb{D}}^{H}_{\varphi} \beta$$

If β depends upon A given $\varphi 1$, and if $\varphi 2$ permits more variety in the system than does $\varphi 1$, there is more opportunity for information transmission, thus β should depend upon A given $\varphi 2$ as well. Formally

Theorem 2-3]

$$\varphi 1 \subseteq \varphi 2 \quad \supset. \quad A \mathrel{\mathbb{D}}^{H}_{\varphi 1} \beta \supset A \mathrel{\mathbb{D}}^{H}_{\varphi 2} \beta$$

(note: $\varphi 1 \subseteq \varphi 2 \quad \equiv_{def} \quad (\forall \sigma)(\varphi 1(\sigma) \supset \varphi 2(\sigma))$

----- Section 2.5 --- Reflexivity [rflx:]

In this section we explore the reflexivity of Strong Dependency. We show that it may not be reflexive over execution of some history if that history causes the value of some object to be written over. We show that it is not reflexive over the empty history if some object initially exhibits no variety.

Strong Dependency may be reflexive. Consider a system in which both α and β are 16 bit integers and

δ: $\beta \leftarrow \alpha$

We find that $\alpha \mathbb{D}^{\delta} \alpha$. All of the variety initially in α remains in α after execution of δ.

However, $\neg\ \beta \mathbb{D}^{\delta} \beta$. Over execution of δ, the original contents of β are destroyed, so that any variety among possible initial values of β will not be retained in (or conveyed to) β after execution of δ. In fact, any of the initial variety in β is completely lost to the system.

Dependency is generally reflexive over the empty history, except where an object is constrained so that it may only contain a single value. If

$\varphi(\sigma) \equiv \sigma.\alpha = 37$

we find that $\alpha \mathbb{D}^{\lambda} \alpha$ but $\neg\ \alpha \mathbb{D}^{\lambda}_{\varphi} \alpha$

The constraint φ eliminates any of the variety in α. If α does admit variety, then certainly that variety will not be destroyed over the empty history. But if α is constrained so that no variety is there initially, the empty history will not convey any new variety to α.

If φ eliminates the variety in some set A, then no information can be transmitted from A to any object over any history. If there is no variety in A, there is none to be conveyed. Formally,

Theorem 2-4]

$$(\forall\alpha\lambda A)[\ \neg\ A \mathbb{D}^{\lambda}_{\varphi} \alpha\ ) \supset (\forall\beta)[\ \neg\ A \mathbb{D}_{\varphi} \beta\ )$$

Finally we note that any information transmission over the empty history must be reflexive. If no operation is executed, no real (non-reflexive) information transmission can take place.

Theorem 2-5]

$$A \mathbb{D}^{\lambda}_{\varphi} \beta \supset \beta\ \lambda\ A$$

----- Section 2.6 --- Autonomy [autonomy:]

In this section, we discuss a class of constraints on the initial state we call autonomous. In chapter 5, we show that the Strong Dependency formalism corresponds to our intuitive notion of information transmission for autonomous constraints, whereas it may not for non-autonomous constraints.

Autonomous constraints restrict the variety in each object independently of the values of other objects. Non-autonomous constraints indicate relationships among the values of different objects. For example,

$\varphi(\sigma) \equiv \sigma.\alpha \leq 10 \wedge (\sigma.\beta \equiv 6 \bmod 11)$ is autonomous
$\varphi(\sigma) \equiv \sigma.\alpha \leq 10 \wedge \sigma.\beta \leq 10$ is autonomous
$\varphi(\sigma) \equiv (\forall x)(\ \sigma.x \leq 10\ )$ is autonomous

$\varphi(\sigma) \equiv \sigma.\beta = \sigma.\alpha + 10$ is non-autonomous
$\varphi(\sigma) \equiv \sigma.\alpha \leq 10 \supset \sigma.\beta = 4$ is non-autonomous

For now, we can think of autonomous constraints as a conjunction of conditions, each condition independently constraining the value of a single object. A formal definition of autonomy can be found in section 5.4.

Though autonomy seems quite a strict condition, it does model a number of common useful situations. For example, in [Cohen 76], we consider the problem of guaranteeing that a set of "sensitive" objects can only be altered by certain processes executing verified programs. The initial constraint on the protection state that guaranteed that the condition held was quite complex, but autonomous nonetheless.

Autonomous predicates are useful for "typing" objecs. One might partition objects on some basis. For example, Int(x) might be true if x were to represent an integer, while Smallint(x) might characterize small integers. An autonomous φ might then require that objects representing small integers have small integer values. Formally

$\varphi(\sigma) \equiv (\forall x)(\ \text{Smallint}(x) \supset -16 \leq \sigma.x \leq 15\ )$

Alternately, each object might itself contain a designation of its own

type as well as it's value. The corresponding autonomous constraint might then be:

$$\varphi(\sigma) \equiv (\forall x)[\ \sigma.x.\text{type} = \text{"smallint"} \supset -16 \leq \sigma.x.\text{value} \leq 15\ ]$$

The consideration of non-autonomous constraints adds a certain complexity to the analysis of information transmission. As we noted above, Strong Dependency does not necessarily correspond to information transmission for non-autonomous constraints.

In section 2.3, we extended the formalism of strong dependency to allow the source of information transmission to be a set of objects. We showed that, if information is transmitted from a set of object A to β, then at least one of the objects in A must itself be a source. This remains true even if we autonomously constrain the system.

Theorem 2-6]

If φ is autonomous then

$$A \mathbb{D}^{H}_{\varphi} \beta \supset (\exists \alpha \in A)[\ \alpha \mathbb{D}^{H}_{\varphi} \beta\ ]$$

Chapter 3 - Solving Information Problems [infsolv]

----- Section 3.1 --- Introduction []

> "...the subject matter of Cybernetics is not events or objects but the information "carried" by events and objects. We consider the objects or events only as proposing facts, propositions, messages, precepts, and the like."
>
> Gregory Bateson "Cybernetic Explanation"

In this chapter we discuss information problems, problems concerned with preventing information transmission in computational systems. Using the Strong Dependency formalism, we define two well known information problems, the Confinement Problem and the Security Problem.

We discuss maximal solutions and consider information transmission as a criteria for evaluating and comparing solutions to problems.

----- Section 3.2 --- Constraint as Solution [consol:]

In [Cohen 76], we argue that problems in computational systems can be solved by finding a way to constrain the states in which the system is initially permitted to operate. We characterize appropriate initial constraints by a predicate χ. For example, the solutions to the enforcement problem $\Psi_{problem}$ (section 1.4) can be characterized by

$$\chi(\varphi) \equiv \varphi \text{ enforces } \Psi_{problem}$$

If $A \not\rhd \beta$ and $\neg A \not\rhd_{\varphi} \beta$, then φ can be viewed as a solution to the following problem: Find a way to guarantee that no information is transmitted from A to β. The solutions to this problem may be defined by

$$\chi(\varphi) \equiv \neg A \not\rhd_{\varphi} \beta$$

Suppose we wanted to guarantee that no information could be transmitted from α to β in the system

δ: if m then $\beta \leftarrow \alpha$

The obvious solution to the problem is

$$\varphi(\sigma) \equiv \neg\sigma.m$$

for by initially constraining states to those in which m is false, we guarantee that execution of δ will have no effect on β. However there is another solution

$$\varphi(\sigma) \equiv \sigma.\alpha = 13$$

By constraining α to be 13, no variety remains in α and none can therefore be transmitted to β. We can, if we so choose, eliminate such solutions by requiring that φ be independent of α, that is, by requiring that the value of α have no effect upon the truth of φ. Formally we can define

>> Def 3-1] φ is A-independent iff

$$(\forall\sigma 1\sigma 2)\,(\ \sigma 1 \underset{A}{\equiv} \sigma 2 \supset .\ \varphi(\sigma 1) \equiv \varphi(\sigma 2)\ )$$

The problem of guaranteeing that no information is transmitted from α to β can then be redefined as

$$X(\varphi) \equiv \neg\ \alpha \ \mathbb{D}_{\varphi}\ \beta \ \wedge\ \varphi \text{ is } \alpha\text{-independent}$$

----- Section 3.3 --- Initial and Invariant Constraints [infinv:]

In this section, we explore the difference between invariant and non-invariant constraints.

When we describe a problem as $X(\varphi)$, φ only represents an initial constraint, not necessarily an invariant one. Likewise, when we indicate that some constraint on the variety in an object may prevent information transmission, that constraint is just an initial constraint as well (section 2.4). Consider the problem

$$\chi(\varphi) \equiv \neg\ \alpha \mathrel{\mathbb{D}_{\varphi}} \beta$$

in the system

δ1: if flag then $\beta \leftarrow \alpha$ else $\beta \leftarrow \emptyset$
δ2: (flag ← tt; $\alpha \leftarrow x$)

A solution to this problem (that is α-independent as well) is

$$\varphi(\sigma) \equiv \neg\sigma.\mathrm{flag}$$

If flag is false, then execution of δ1 does not transmit information from α to β; it always sets β to ∅. However φ is not invariant. Execution of δ2 sets flag to true. Subsequent execution of δ1 would transmit information from α to β. Nonetheless φ is a solution, for execution of δ2 also destroys the information initially contained in α by overwriting it with x. So while subsequent execution of δ1 will permit β to reflect the most recent value of α, it reflects nothing of α's initial value.

Hence, if φ is a solution to the problem

$$\chi(\varphi) \equiv \neg\ \alpha \mathrel{\mathbb{D}_{\varphi}} \beta$$

then φ, in general, is only an initial but not invariant constraint and guarantees only that no information initially contained in α can be transmitted to β. Values placed in α after execution of some history may have an effect on the value of β.

----- Section 3.4 --- Examples of Information Problems [xmplinfo:]

A simple version of the Confinement Problem [Lampson 73] can now be stated. Suppose that Confined(x) if x is the name of an object initially containing information that is to be confined. Suppose that Spy(x) if x names an object to which this confined information must not be transmitted. We can define the Confinement Problem as (also see section 7.5)

$$\chi(\varphi) \equiv (\forall\alpha,\beta)(\ \alpha \mathrel{\mathbb{D}_{\varphi}} \beta\ \supset.\ \mathrm{Confined}(\alpha) \supset \neg\mathrm{Spy}(\beta)\ )$$

That is, find some constraint φ that reduces information transmission in the system so that, if information is transmitted from α to β, and α is confined, then β must not be a spy.

A solution to the Security Problem [Case 74] would guarantee that information is never transmitted from one object to a second object at a lower security classification than the first. We can define the Security Problem as

$$\chi(\varphi) \equiv (\forall \alpha, \beta)(\alpha \not\rhd_{\varphi} \beta \supset. \ Cls(\alpha) \leq Cls(\beta))$$

where $Cls(x)$ is the classification of x. In [Case 74], φ is referred to as the requirement for a "secure system".

[Note that as in [Denning 75], the classification need not be a single value, but could be a vector of clearance/classification values, in which case "$\leq$" would describe a partial rather than a total order.]

----- Section 3.5 --- Maximal Solutions [maxsol:]

We say a solution is maximal if it is less restrictive (allows more initial states) than any other solution. Information problems do not necessarily have unique maximal solutions.

A maximal solution for a problem is unique if the problem can be shown to satisfy the Join property [Cohen 76]. That is, if

$$\chi(\varphi 1) \wedge \chi(\varphi 2) \supset \chi(\varphi 1 \vee \varphi 2)$$

for then the maximal solution would be the join of all the solutions

$$\varphi_{max} \equiv \vee \{ \varphi \mid \chi(\varphi) \}$$

However, solutions to information problems do not satisfy the join property. For example, consider the problem

$$\chi(\varphi) \equiv \backslash\, \alpha \,\mathbb{D}_{\varphi}\, \beta$$

in the system

$$\delta: \quad \underline{\text{if}}\ m \ \ \underline{\text{then}}\ \beta \leftarrow \alpha$$

One solution to χ is

$$\varphi 1(\sigma) \equiv \sigma.\alpha = 13$$

If α is constrained to be a constant, no information is transmitted to β. Any constant will do, so another solution is

$$\varphi 2(\sigma) \equiv \sigma.\alpha = 74$$

However, the join of these solutions

$$(\varphi 1 \vee \varphi 2)(\sigma) \equiv \sigma.\alpha = 13 \vee \sigma.\alpha = 74$$

is <u>not</u> a solution, for $(\varphi 1 \vee \varphi 2)$ <u>does</u> allow variety in α to be transmitted to β by execution of δ. Since the join property does not hold, problems may not have unique maximal solutions. Consider the system

$$\delta: \quad \underline{\text{if}}\ \alpha \leq 10 \ \ \underline{\text{then}}\ \beta \leftarrow 0 \ \ \underline{\text{else}}\ \beta \leftarrow 1$$

The problem $\chi(\varphi) \equiv \neg\, \alpha \,\mathbb{D}_{\varphi}\, \beta$ is solved by both $\varphi 1$ and $\varphi 2$, where

$$\varphi 1(\sigma) \equiv \sigma.\alpha = 6$$
$$\varphi 2(\sigma) \equiv 8 \leq \sigma.\alpha \leq 10$$

A maximal solution containing both of these solutions is

$$\varphi_{max}(\sigma) \equiv \sigma.\alpha \leq 10$$

A <u>different</u> maximal solution is

$$\varphi_{max}(\sigma) \equiv \sigma.\alpha > 10$$

In neither case can a less restrictive solution to χ be found. In both cases, φ_{max} solves χ by guaranteeing that the resulting value in β after execution of δ is always the same. It is always 0 for the first maximal solution, and it is always 1 for the second maximal solution.

By requiring independence (definition 3-1), we can formalize problems whose solutions do satisfy the join property and therefore have unique maximal solutions.

Theorem 3-1]

$$\text{If} \quad \chi(\varphi) \equiv \neg\, A \not\rhd_{\varphi} \beta \quad \wedge \quad \varphi \text{ is A-independent}$$

$$\text{then} \quad \chi(\varphi 1) \wedge \chi(\varphi 2) \supset \chi(\varphi 1 \vee \varphi 2)$$

Consider the system (see section 1.3)

δ: if $s \in \langle x,x \rangle \wedge r \in \langle x,\alpha \rangle \wedge w \in \langle x,\beta \rangle$
then $\beta \leftarrow \alpha$

There is a single maximal solution to the problem

$$\chi(\varphi) \equiv \neg\, \alpha \not\rhd_{\varphi} \beta \wedge \varphi \text{ is } \alpha\text{-independent}$$

It is

$$\varphi_{max}(\sigma) \equiv s \notin \langle x,x \rangle(\sigma) \vee r \notin \langle x,\alpha \rangle(\sigma) \vee w \notin \langle x,\beta \rangle(\sigma)$$

----- Section 3.6 --- Comparing Solutions [infsuf:]

"Variety, within the limits of satisfactory constraints, may be a desirable end in itself..."

Herb Simon "The Sciences of the Artificial"

In [Cohen 76], we argue that solutions to problems should, in general, be as unrestrictive as possible. That is, one should strive to obtain maximal

solutions to problems. Yet in many cases, solutions that are not maximal may be as good in certain respects as those that are maximal. We would like to find measures that characterize the worth of a solution, which would indicate that certain non-maximal solutions are as worthy as those which are maximal. In this section, we will show that Strong Dependency may be an appropriate base for just such a measure.

Consider the problem

$$\chi(\varphi) \equiv \neg\ \alpha \Rrightarrow_{\varphi} \beta \ \wedge\ \varphi \text{ is } \alpha\text{-independent}$$

in the system (see section 1.3)

$\delta 1$: <u>if</u> $s \in \langle x,x\rangle \wedge r \in \langle x,\alpha\rangle \wedge w \in \langle x,\beta\rangle$
<u>then</u> $\beta \leftarrow \alpha$

$\delta 2$: <u>if</u> $s \in \langle x,x\rangle \wedge r \in \langle x,m\rangle \wedge w \in \langle x,\beta\rangle$
<u>then</u> $\beta \leftarrow m$

A maximal solution is (see section 3.5)

$$\varphi_{max}(\sigma) \equiv s \notin \langle x,x\rangle(\sigma) \vee r \notin \langle x,\alpha\rangle(\sigma) \vee w \notin \langle x,\beta\rangle(\sigma)$$

Another solution (more restrictive than φ_{max}) is

$$\varphi 1(\sigma) \equiv r \notin \langle x,a\rangle(\sigma)$$

While $\varphi 1$ is stricter than φ_{max}, the two share an important property. They prevent information transmission from α to β but prevent no other information transmission (for example from m to β). Contrast those solutions with the solution (also contained in φ_{max})

$$\varphi 2(\sigma) \equiv s \notin \langle x,x\rangle(\sigma) \vee w \notin \langle x,\beta\rangle(\sigma)$$

which prevents information transmission from x to β as well. We will develop a criteria that indicates that $\varphi 1$ is as worthy a solution for χ as φ_{max} while $\varphi 2$ is not, by formalizing the determination of which information paths are eliminated.

First, let us take a moment and explore the difficulty of comparing solutions quantitatively. We might argue that one solution is as good as another if it allows more bits of information to be transmitted in the system. Suppose that α, β, t1, t2, m1 and m2 are all 16 bit non-negative integers. Consider the system

$\delta 1$: m1 $\leftarrow$ t1
$\delta 2$: m2 $\leftarrow$ t2 .
$\delta 3$: <u>if</u> t1 $\geq$ 4 $\wedge$ t2 $\geq$ 256 <u>then</u> $\beta \leftarrow \alpha$

The problem

$$\chi(\varphi) \equiv \neg\ \alpha \not\rhd_{\varphi} \beta$$

can be solved by either $\varphi 1$ or $\varphi 2$ where

$$\varphi 1(\sigma) \equiv \sigma.t1 \leq 3$$
$$\varphi 2(\sigma) \equiv \sigma.t2 \leq 255$$

We might think that $\varphi 2$ is a better solution since it only reduces t2's variety to 8 bits while $\varphi 1$ reduces t1's variety to 2 bits worth. This kind of analysis is uncomfortable for a number of reasons. First, numeric values give no sense of the relative importance of the information in t1 and t2. Secondly, to formally assign a bit value to the amount of information transmitted we really need to know the probability of each initial state and the probability of each behavior in the system (see section 7.4).

We opt for a qualitative rather than quantitative measure of worth. We will measure the worth of a solution in terms of whether or not information can be transmitted at all. Formally, we define the worth of a solution as the set of information paths permitted in the system when constrained by the solution.

$$Worth(\varphi) \equiv \{ \langle A,\beta \rangle \mid A \not\rhd_{\varphi} \beta \}$$

If we order these worths by whether one is a subset of the other, then we find that

$$\text{Worth}(\varphi 1) \leq \text{Worth}(\varphi 2) \quad \underline{\text{iff}}$$

$$(\forall A, \beta)\,(\; A \mathrel{\mathbb{D}}_{\varphi 1} \beta \;\supset\; A \mathrel{\mathbb{D}}_{\varphi 2} \beta \;)$$

In [Cohen 76], we note that measures of worth should ordinarily be monotonic. Formally

>> Def 3-2] <Worth,≤> is a monotonic measure iff

$$\varphi 1 \subseteq \varphi 2 \;\supset.\; \text{Worth}(\varphi 1) \leq \text{Worth}(\varphi 2)$$

That is, if one solution to a problem is less restrictive than another, it should be at least as worthy. We show in [Cohen 76] that if a problem has a unique maximal solution, that it is the worthiest solution relative to any monotonic measure. From theorem 2-3, it is clear that this measure of worth is a monotonic one.

According to the measure of worth we have defined, two solutions are equally worthy if neither eliminates an information path permitted by the other.

Chapter 4 - Strong Dependency Induction [strind:]

----- Section 4.1 --- Introduction []

In this chapter, we discuss Strong Dependency Induction, an inductive proof technique for proving the correctness of solutions to information problems. We confine our attention to solutions which are both autonomous and invariant, treating more general cases in chapters 5 and 6.

We find that Strong Dependency Induction is not useful unless the Strong Dependency relation is transitive. We introduce another technique, which we call Separation of Variety, in order to extend Strong Dependency Induction to the non-transitive case.

----- Section 4.2 --- Transmission Through Intermediate Objects [invar:]

The reader might imagine that if information is transmitted from α to β by $\delta 1 \delta 2$ in some system, there should be some intermediate object m (possibly the same as α or β in degenerate cases) such that $\delta 1$ transmits information from α to m and $\delta 2$ transmits information from m to β. For example, in the system

$\delta 1$: $m \leftarrow \alpha$
$\delta 2$: $\beta \leftarrow m$

$\alpha \mathbb{D}^{\delta 1 \delta 2} \beta$ and $\alpha \mathbb{D}^{\delta 1} m$ and $m \mathbb{D}^{\delta 2} \beta$

This intuition is exactly right and holds more generally when the system is initially constrained by an autonomous invariant constraint.

Theorem 4-1]

If φ is autonomous and invariant then

$$\alpha \mathbb{D}_{\varphi}^{HH'} \beta \supset (\exists m)(\alpha \mathbb{D}_{\varphi}^{H} m \wedge m \mathbb{D}_{\varphi}^{H'} \beta)$$

This is the basic induction theorem for information transmission, although in later sections we will develop more general proof techniques. We will find the following corollaries useful:

Corollary 4-2]

If φ is autonomous and invariant and $\alpha \neq \beta$ then

$$(\forall m \neq \alpha, \delta)(\neg\, \alpha \mathbb{D}^{\delta}_{\varphi} m) \vee (\forall m \neq \beta, \delta)(\neg\, m \mathbb{D}^{\delta}_{\varphi} \beta) \supset .\ \neg\, \alpha \mathbb{D}_{\varphi} \beta$$

That is, if either no operation can transmit information from α to any other object or no operation can transmit information from any other object to β, then information cannot be transmitted from α to β. Another useful corollary is

Corollary 4-3]

If φ is autonomous and invariant
and q is reflexive and transitive
(q reflexive - $(\forall x)(q(x,x))$
q transitive - $q(x,y) \wedge q(y,z) \supset . q(x,z)$)

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

$$\supset .\ (\forall x, y)(x \mathbb{D}_{\varphi} y \supset q(x,y))$$

Using these corollaries, we need only analyze information transmission over the set of all operations rather than over the set of all histories.

The last corollary is especially useful for the Security Problem (section 3.4) which requires a solution guaranteeing that whenever information is transmitted from α to β, β's classification must be no less than α's. The problem can be formally stated as

$$X(\varphi) \equiv (\forall \alpha, \beta)(\alpha \mathbb{D}_{\varphi} \beta \supset .\ Cls(\alpha) \leq Cls(\beta))$$

where $Cls(x)$ is the classification of x.

$q(x,y) \equiv Cls(x) \leq Cls(y)$ is an example of a transitive, reflexive q. By corollary 4-3, if Φ is autonomous and invariant, we only need show that no operation can transmit information from one object to another at a lower classification (when the system is constrained by Φ) to show that the system is secure. That is, we need only show that

$$(\forall \delta, \alpha, \beta)\ (\ \alpha \not\rhd^{\delta}_{\Phi} \beta \ \supset .\ Cls(\alpha) \leq Cls(\beta)\ )$$

When $\alpha \rhd^{\delta} \beta$ is read as "information flows from α to β over execution of δ", this corollary provides a formal basis for the work discussed in [Denning 75] that describes information flow in systems where objects have statically assigned classifications.

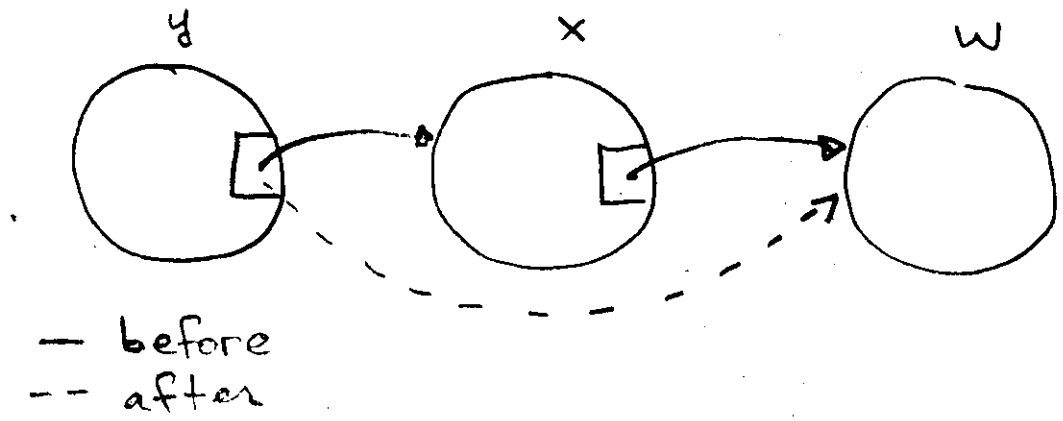
----- Section 4.3 --- An Example of Strong Dependency Induction [invxmpl:]

In this section, we will present a detailed example showing how Strong Dependency Induction may be used to solve an information problem.

Imagine a system where each object contains data as well as a single pointer to another object: The system has two sets of operations:

```
δ1(y,x):  if y.ptr = x  then y.data ← x.data
δ2(y,x):  if y.ptr = x  then y.ptr ← x.ptr
```

If y points to x, then execution of δ1(y,x) will copy data from x to y. If y points to x and x points to w, then after execution of δ2(y,x), y will point to w, as illustrated below.

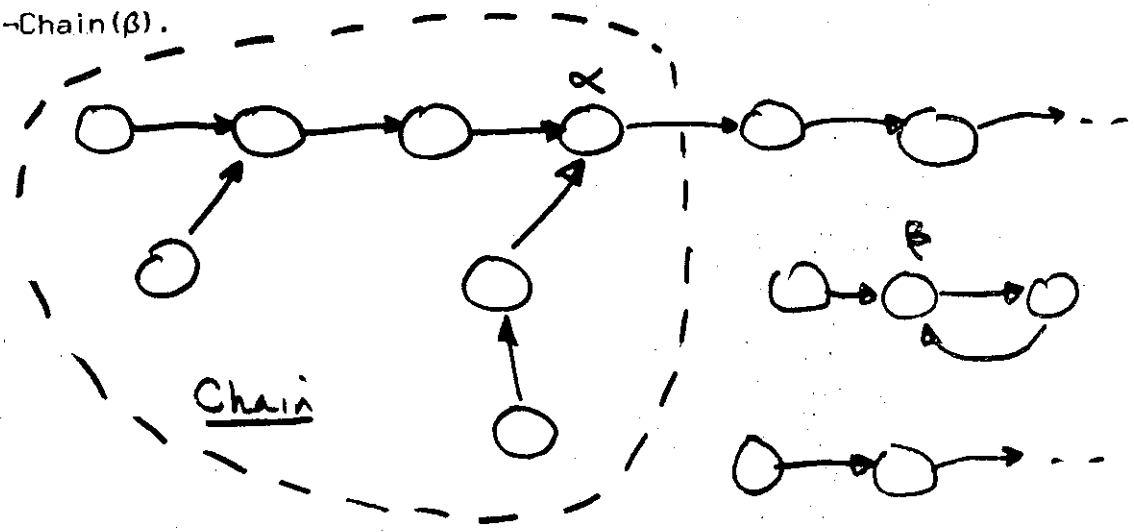


We will consider the problem of trying to guarantee that information from a particular object α cannot be transmitted to some other object β. That is:

$$\chi(\varphi) \equiv \neg\, \alpha \, \mathbb{D}_{\varphi} \, \beta$$

We will show that if there is no chain of pointers from β to α, then no information can be transmitted from α to β.

We divide the objects into two sets, those that point through some chain of objects (possibly of length zero) to α and those that do not. We will characterize the former by the predicate Chain, so that Chain(α) and $\neg$Chain(β).



We argue that if β does not initially point to α, then no information can be transmitted from α to β. First we show that the initial constraint φ guarantees that β does not point to α, where φ is

$$\varphi(\sigma) \equiv (\forall y)\,(\,\mathrm{Chain}(\sigma, y.\mathrm{ptr}) \supset \mathrm{Chain}(y)\,)$$

We will write points(y,x,n)(σ) to mean that there is a chain of pointers of length n from y to x in state σ. Formally, we can define points recursively as

$$points(y,x,0)(\sigma) \iff y = x$$

$$points(y,x,n+1)(\sigma) \iff (\exists m)(\sigma.y.ptr = m \wedge points(m,x,n)(\sigma))$$

A straightforward induction on n shows that

$$\varphi(\sigma) \supset (\forall n)(points(y,x,n)(\sigma) \supset . Chain(x) \supset Chain(y))$$

Since $Chain(\alpha)$ and $\neg Chain(\beta)$, we conclude that

$$\varphi(\sigma) \supset (\forall n)(\neg points(\beta,\alpha,n)(\sigma))$$

That is, φ guarantees that β does not point to α.

φ is autonomous. We next show that φ is invariant. $\delta 1$ has no effect on pointers, so we need only consider $\delta 2$. Given that $\varphi(\sigma)$ holds, we will show that for arbitrary p and q, $\varphi(\delta 2(q,p)(\sigma))$ holds.

1] Given $\varphi(\sigma)$
2] Assume $Chain(\delta 2(q,p)(\sigma).y.ptr)$
3] Case 1 $y \neq q$
4] $Chain(\sigma.y.ptr)$ [2,3, Def $\delta 2$]
5] Case 2 $y = q$, $\sigma.y.ptr \neq p$
6] $Chain (\sigma.y.ptr)$ [2,5, Def $\delta 2$]
7] Case 3 $y = q$, $\sigma.y.ptr = p$
8] $Chain(\sigma.p.ptr)$ [2,7, Def $\delta 2$]
9] $Chain(p)$ [1,8]
10] $Chain(\sigma.y.ptr)$ [7,9]
11] $Chain(\sigma.y.ptr)$ [3-4,5-6,7-10]
12] $Chain (y)$ [11,1]
13] $\varphi(\delta 2(q,p)(\sigma))$ [2-12]

Next we pick

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

noting that q is both reflexive and transitive. We next show that

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

1] Assume $x \mathbb{D}^{\delta}_{\varphi} y$ where $\delta \equiv \delta 1(q,p)$

2] Assume Chain(x)

3] $(\exists \sigma 1, \sigma 2)\,(\ \sigma 1 \overset{\varphi}{\underset{x}{\Diamond}}\overset{\delta}{\underset{y}{}} \sigma 2\ )$ [1]

4] $\sigma 1 \underset{x}{=} \sigma 2 \ \wedge\ \delta(\sigma 1).y \neq \delta(\sigma 2).y$ [3]

5] $(\ \sigma 1.y.ptr = x \ \vee\ \sigma 2.y.ptr = x\ )$ [4, 1(def of δ)]

6] Chain($\sigma 1.y.ptr$) $\vee$ Chain($\sigma 2.y.ptr$) [5,2]

7] $\varphi(\sigma 1) \ \wedge\ \varphi(\sigma 2)$ [3]

8] Chain(y) [6,7]

The proof for $\delta 2$ is exactly the same. Since q is transitive and reflexive, and φ is autonomous and invariant, by corollary 4-3, we can show that

$(\forall x, y)\,(\ x \mathbb{D}_{\varphi} y \supset q(x,y)\ )$

Since Chain(α) and $\neg$Chain(β), this result shows (see the definition of q above) that

$\neg\ \alpha \mathbb{D}_{\varphi} \beta$

This shows that φ is a solution to χ. If there is no chain of pointers from β to α, then information cannot be transmitted from α to β.

----- Section 4.4 --- Transitivity [trans:]

In this section, we show that the useful application of Strong Dependency Induction requires that Strong Dependency be transitive.

In the system

$\delta 1$: if q then $m \leftarrow \alpha$

$\delta 2$: if $\neg q$ then $\beta \leftarrow m$

We can show directly that the problem

$$\chi(\varphi) \equiv \neg\, \alpha \, \mathbb{D}_{\varphi}^{\delta 1 \delta 2} \, \beta$$

can be solved by the always true solution, $\varphi(\sigma) \equiv tt$. That is, for any two states initially differing only in α, after execution of $\delta 1 \delta 2$, the value of β will be the same for both.

However, to prove this result by Strong Dependency Induction, we would have to show that either

$$\neg\, \alpha \, \mathbb{D}^{\delta 1} \, m \qquad \text{or} \qquad \neg\, m \, \mathbb{D}^{\delta 2} \, \beta$$

But both

$$\alpha \, \mathbb{D}^{\delta 1} \, m \qquad \text{and} \qquad m \, \mathbb{D}^{\delta 2} \, \beta$$

The difficulty is that Strong Dependency is not transitive in this system. Strong Dependency is transitive if

$$\alpha \, \mathbb{D}^{H} \, m \;\wedge\; m \, \mathbb{D}^{H'} \, \beta \;\supset.\; \alpha \, \mathbb{D}^{HH'} \, \beta$$

----- Section 4.5 --- Separation of Variety [sepvar:]

In this section we introduce a proof technique we call Separation of Variety which can be used to extend Strong Dependency Induction to cases where Strong Dependency is not transitive. We explain Separation of Variety by considering the system

δ: if α then $\beta \leftarrow tt$ else $\beta \leftarrow ff$

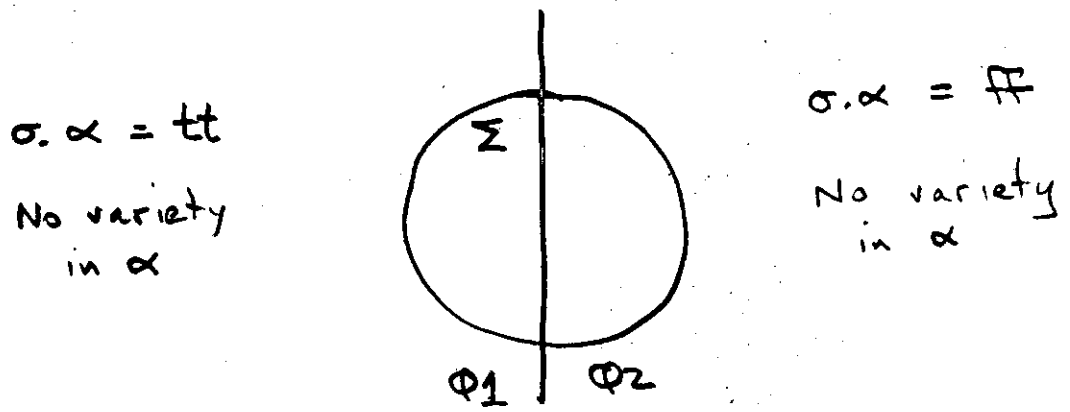
While $\alpha \, \mathbb{D}^{\delta} \, \beta$, there are two solutions, $\varphi 1$ and $\varphi 2$, to the problem

$$\chi(\varphi) \equiv \neg\, \alpha \, \mathbb{D}_{\varphi} \, \beta$$

$$\varphi 1(\sigma) \equiv \sigma.\alpha = tt$$
$$\varphi 2(\sigma) \equiv \sigma.\alpha = ff$$

In both solutions, we prevent transmission by reducing the variety of α.

We can think of $\varphi 1$ and $\varphi 2$ as covering the state space, Σ, as illustrated in the diagram below.

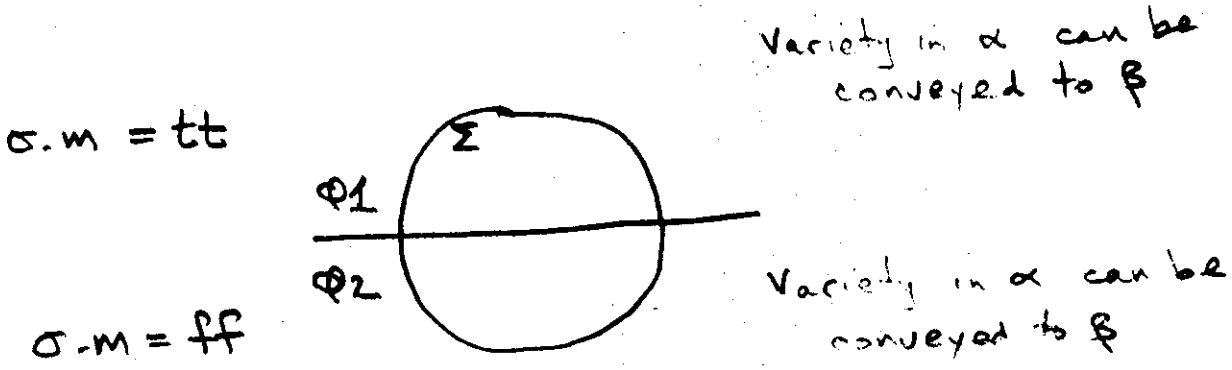


While the set of <u>all</u> possible states do exhibit variety in α, $\varphi 1$ and $\varphi 2$ <u>separate</u> that <u>variety</u> and prevent information transmission. [In effect, this is the reason why the join property does not hold for information problems (section 3.5).]

But what would happen if $\varphi 1$ and $\varphi 2$ separated the variety in some other object instead of α? For example,

$$\varphi 1(\sigma) \equiv \sigma.m = tt$$
$$\varphi 2(\sigma) \equiv \sigma.m = ff$$

Given either $\varphi 1$ or $\varphi 2$ as a constraint, transmission can still take place. Both $\alpha \rhd_{\varphi 1} \beta$ and $\alpha \rhd_{\varphi 2} \beta$.

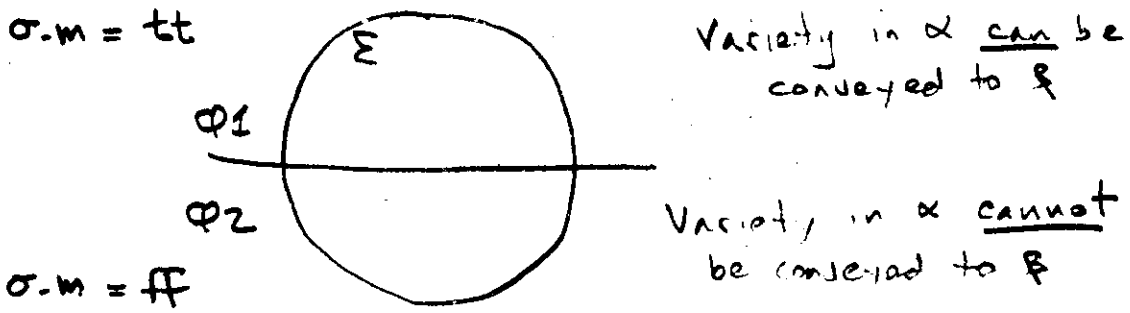


Each subset of Σ characterized by $\varphi 1$ or $\varphi 2$ still exhibits all of the variety possible in α - and in each case, all of that variety can still be transmitted to β. Now, let's consider the system

δ: <u>if</u> m <u>then</u> $\beta \leftarrow \alpha$

We find that

$$\alpha \; \mathbb{D}^{\delta}_{\varphi 1} \; \beta \qquad \text{but} \qquad \neg \; \alpha \; \mathbb{D}^{\delta}_{\varphi 2} \; \beta$$



While σ's variety is still completely exhibited in both subsets of Σ characterized by $\varphi 1$ and $\varphi 2$, $\varphi 1$ prevents that variety from being conveyed to β. However, in the case of $\varphi 2$, transmission can still take place.

In general, if we split the state space in any way along partitions <u>independent</u> of α, in at least one of the cases distinguished by the split, α's variety can still be conveyed to β. If not, there would have been no way that α's variety <u>ever</u> could have been transmitted to β. In fact, the result holds for a more general sort of division of the state space. If $\varphi 1, \ldots, \varphi n$ cover Σ along lines independent of α, then $\alpha \; \mathbb{D}_{\varphi i} \; \beta$ for at least one of the i's. We defined independence (definition 3-1) so that

φ is A-independent <u>iff</u>

$$(\forall \sigma, \sigma') \; (\; \sigma \underset{A}{=} \sigma' \supset . \; \varphi(\sigma) \equiv \varphi(\sigma') \;)$$

That is, φ is A-independent if φ in no way constrains the value of any

object in A. Next we define an A-independent cover, a set of A-independent constraints that cover Σ.

>> Def 4-1] $\{ \varphi_i \}$ is an A-independent cover iff

$$(\forall i)(\varphi_i \text{ is A-independent }) \quad \wedge$$
$$(\forall \sigma \exists i)(\varphi_i(\sigma))$$

Theorem 4-4]

If $\{ \varphi_i \}$ is an α-independent cover then

$$\alpha \; \mathbb{D}^{H} \; \beta \quad \supset \quad (\exists i)(\alpha \; \mathbb{D}^{H}_{\varphi_i} \; \beta)$$

and therefore

$$\alpha \; \mathbb{D} \; \beta \quad \supset \quad (\exists i)(\alpha \; \mathbb{D}_{\varphi_i} \; \beta)$$

More generally

Theorem 4-5]

If $\{ \varphi_i \}$ is an A-independent cover then

$$A \; \mathbb{D}^{H}_{\varphi} \; \beta \quad \supset \quad (\exists i)(A \; \mathbb{D}^{H}_{\varphi \wedge \varphi_i} \; \beta)$$

and therefore

$$A \; \mathbb{D}_{\varphi} \; \beta \quad \supset \quad (\exists i)(A \; \mathbb{D}_{\varphi \wedge \varphi_i} \; \beta)$$

[Note that this theorem does not require that the φ_i's be autonomous, only A-independent.]

The theorem suggests the following proof technique. To show

$$\neg \; \alpha \; \mathbb{D}_{\varphi} \; \beta$$

find an α-independent cover $\{ \varphi_i \}$ and show that

$$(\forall i)\ (\ \neg\ \alpha \not\rhd_{\varphi \wedge \varphi i}\ \beta\ )$$

----- Section 4.6 --- An Example of Separation of Variety [sepxmpl:]

We will illustrate the use of separation of variety (in conjunction with Strong Dependency Induction) by showing that $\neg\ \alpha \not\rhd \beta$ in the system

$\delta 1$: if q then $m \leftarrow \alpha$
$\delta 2$: if $\neg q$ then $\beta \leftarrow m$

Pick the α-independent cover $\{\varphi 1, \varphi 2\}$, where

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

We find that

$$(\forall m \neq \beta, \delta)\ (\ \neg\ m \not\rhd^{\delta}_{\varphi 1}\ \beta\ )$$

so by corollary 4-2, $\neg\ m \not\rhd_{\varphi 1} \beta$. Similarly

$$(\forall m \neq \alpha, \delta)\ (\ \neg\ \alpha \not\rhd^{\delta}_{\varphi 2}\ m\ )$$

so by corollary 4-2, $\neg\ \alpha \not\rhd_{\varphi 2} m$

Therefore, by theorem 4-5, $\neg\ \alpha \not\rhd \beta$

For another example, consider the system ("left" and "right" are assumed to be disjoint components of m)

$\delta 1$: $m.left \leftarrow \alpha$
$\delta 2$: $\beta \leftarrow m.right$

We pick $\varphi i(\sigma) \equiv \sigma.m.right = i$

We must show that for each φi, no information can be transmitted from α to β given φi. We will prove this for each φi using corollary 4-2. This requires

a proof that each Φi is invariant and that given Φi, no operation can transmit information to β from any other object. Each Φi is invariant since $\delta 2$ does not modify m and $\delta 1$ only modifies m.left. Now, though $\delta 2$ modifies β by copying the value of m2.right into β, when Φi constrains m2.right to be a constant, no variety is conveyed to β and thus no information is transmitted to β. Since $\delta 1$ does not affect β at all, no operation can transmit information to β from any other object. Formally

1) $(\forall i)$ (Φi is α-independent)
2) $(\forall \sigma \exists i)$ ($\Phi i(\sigma)$) [Pick the Φi so that σ.m.right = i]
3) $(\forall i)$ (Φi is autonomous)
4) $(\forall i)$ (Φi is invariant) [left to reader]
5) $(\forall i)(\forall x \neq \beta, \delta)$ ($\neg x \overset{\delta}{\underset{\Phi i}{\triangleright}} \beta$) [left to reader]
6) $(\forall i)$ ($\neg \alpha \underset{\Phi i}{\triangleright} \beta$) [3,4,5, Cnlry 4-2]
7) $\neg \alpha \triangleright \beta$ [1,2,6, Th 4-5]

Chapter 5 - Relatively Autonomous Constraints [relphi:]

----- Section 5.1 --- Introduction []

In the previous chapters, we confined our attention to autonomous constraints so that we could explore the basic properties of Strong Dependency - the transmission and separation of variety, and the definition and solution of information problems using Strong Dependency.

In this chapter, we turn our attention to the meaning of constraint. In an information theoretic sense, constraint has two meanings. We explored the first of these meanings in chapter 2, where we showed how constraint might be used to reduce the variety in a system, thereby preventing information transmission.

Constraint has another meaning as well. Non-autonomous constraints establish relations among the values of two or more objects. As a result, they spread the source of transmitted information. For example, the constraint

$$\varphi(\sigma) \equiv \sigma.\alpha \leq \sigma.m$$

relates the initial values of α and m. If information can be transmitted from m to β, information may be transmitted from α to β as well. If an observer of β can discover something about m's value, then β might discover something about α as well, by knowing the relationship between α and m.

We find in this chapter that the Strong Dependency formalism is not wholly suited to dealing with non-autonomous constraints. [Work in progress (section 7.2) is directed towards that goal.]

We find that we can continue to use Strong Dependency for certain non-autonomous constraints. If we "clump" a group of objects together and treat them as a "pseudo-object", then a non-autonomous constraint may appear to be autonomous with respect to that "pseudo-object". For example, if we clump α and m together, then we note that φ is autonomous relative to the clump $\{\alpha,m\}$.

We treat clumps formally as sets, call the related constraints, relatively autonomous, and extend Strong Dependency Induction to handle such constraints.

----- Section 5.2 --- The Strong Dependency Hypothesis [strhyp:]

In this section we show that Strong Dependency is not completely suitable as a formalism for information transmission in systems constrained by non-autonomous constraints.

Strong Dependency represents an attempt to formalize the intuitive notion of information transmission. So far, we accept the following hypothesis.

******* The Strong Dependency Hypothesis *******

If $A \triangleright_{\varphi} \beta$ then

Information can be transmitted from (some object in) A to β in a system constrained initially by φ

In other work (section 7.2), we find additional support for this hypothesis, regardless of whether φ is autonomous or not.

The converse of the Strong Dependency Hypothesis in not true. Consider the problem

$$\chi(\varphi) \equiv \neg\, \alpha 1 \triangleright_{\varphi} \beta$$

in the system

$$\delta: \quad \beta \leftarrow \alpha 1$$

We find that the non-autonomous constraint

$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2$$

will solve the problem. The solution is similar to that of constraining the

value of $\alpha 1$ to that of a constant. Instead, the value of $\alpha 1$ is constrained to be the same as $\alpha 2$. In either case a degree of freedom is removed from the system. Yet, this solution is disturbing, for one might imagine 3 ways that this solution came to pass.

1. $\alpha 1$ was always the same as $\alpha 2$ in the system. Somehow, in initializing the system, the value of $\alpha 1$ was also placed in $\alpha 2$ (or vice-versa). There is still a great deal of variety in $\alpha 1$; only it is shared with $\alpha 2$. Execution of δ will convey all of this variety to β.

2. φ was brought about (produced - see [Cohen 76]) by executing some other operation (not shown) that copied $\alpha 1$ to $\alpha 2$. The argument of [1] above still holds.

3. φ was brought about by some operation that copied $\alpha 2$ to $\alpha 1$, destroying all of the initial variety in $\alpha 1$. However, we are analyzing the system <u>after</u> φ was brought about (after the solution was produced), that is, after the copy. Again, the variety in $\alpha 1$ is matched by the variety of $\alpha 2$, and as in [1] and [2], the problem of preventing information transmission still remains.

This analysis argues that information <u>is</u> transmitted from $\alpha 1$ to β given φ, even though $\neg\ \alpha 1 \triangleright_{\varphi} \beta$. The constraint φ spreads the variety between $\alpha 1$ and $\alpha 2$. Strong Dependency is insensitive to that spreading of variety; it only takes account of the fact that $\alpha 1$ <u>appears</u> to have no variety at all since it is forced to take on the same value as $\alpha 2$.

----- <u>Section</u> 5.3 --- Relative Autonomy [relaut:]

In this section, we show how Strong Dependency may be used with certain non-autonomous constraints, by considering a set of objects as a single source of information.

In the example in the previous section, we considered $\alpha 1$ as a potential information source. The Strong Dependency formalism only analyzed the effect of $\alpha 1$'s variety on β independently of the variety in other objects,

particularly in $\alpha 2$. Yet in that example φ spread $\alpha 1$'s variety to $\alpha 2$. We must therefore treat $\alpha 1$ and $\alpha 2$ <u>together</u> as a source; determining whether their composite variety can be transmitted to β. And in fact, we <u>can</u> show that

$$\{\alpha 1, \alpha 2\} \; \mathbb{D}_{\varphi} \; \beta$$

Though φ is not autonomous, we will say that it is autonomous <u>relative</u> to $\{\alpha 1, \alpha 2\}$, or $\{\alpha 1, \alpha 2\}$-autonomous. That means there are no correlations between $\{\alpha 1, \alpha 2\}$ and any other object (a formal definition is found below). The argument suggests that although the converse to the Strong Dependency Hypothesis is not true, the following weaker version is true.

******* The Relative Autonomy Hypothesis *******

If φ is A-autonomous
and $\neg \; A \; \mathbb{D}_{\varphi} \; \beta$ then

No information can be transmitted from A to β
in a system constrained initially by φ

Additional support for this hypothesis may be found in other work in progress (section 7.2). Consider the system

$$\delta: \; \beta \leftarrow \alpha 1 - \alpha 2$$

$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2$$

We find that $\neg \; \{\alpha 1, \alpha 2\} \; \mathbb{D}_{\varphi} \; \beta$

Because φ is $\{\alpha 1, \alpha 2\}$-autonomous, the hypothesis argues that information is transmitted neither from $\alpha 1$ nor from $\alpha 2$ to β. This is as it should be. Given the constraint φ, execution of δ will always set β to $\emptyset$ regardless of the initial values of $\alpha 1$ and $\alpha 2$ (which must be the same).

If the constraint φ above were

$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2 \; \wedge \; \sigma.m1 = \sigma.m2$$

φ would still be $\{\alpha 1, \alpha 2\}$-autonomous. Though other objects (m1 and m2) are constrained to have correlated values, no value of $\alpha 1$ or $\alpha 2$ is correlated with any of them. Even for these kind of relatively autonomous constraints, the Relative Autonomy Hypothesis holds. As long as no variety is spread between objects in A and objects outside of A, Strong Dependency accurately reflects information transmission.

We can represent relative autonomy formally in the following way:

First, remember that we defined φ is A-independent as (def 3-1)

$$(\forall \sigma 1, \sigma 2)(\ \sigma 1 \underset{A}{\equiv} \sigma 2 \ \supset . \ \varphi(\sigma 1) \equiv \varphi(\sigma 2) \ )$$

>> Def 5-1] φ is A-strict iff

$$(\forall \sigma 1, \sigma 2)(\ \sigma 1.A = \sigma 2.A \ \supset . \ \varphi(\sigma 1) \equiv \varphi(\sigma 2) \ )$$

φ is A-independent if φ does not constrain any objects in A.
φ is A-strict if φ only constrains objects in A.

>> Def 5-2] φ is A-autonomous iff

$\varphi \equiv \varphi 1 \wedge \varphi 2$
for some $\varphi 1$ which is A-strict
and some $\varphi 2$ which is A-independent

For example

$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2 \wedge \sigma.m1 = \sigma.m2$$

$\varphi 1(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2$ is $\{\alpha 1, \alpha 2\}$-strict
$\varphi 2(\sigma) \equiv \sigma.m1 = \sigma.m2$ is $\{\alpha 1, \alpha 2\}$-independent

Therefore, $\varphi 1 \wedge \varphi 2$ is $\{\alpha 1, \alpha 2\}$-autonomous.

----- Section 5.4 --- Substitution and Autonomy [subaut:]

In this section, we present a different characterization of relatively autonomous constraints. We show it is equivalent to the definition of relative autonomy given in the previous section, but leads to a more usable formalism. We also define autonomy as it has been used since section 2.6.

Imagine two states $\sigma 1$ and $\sigma 2$ that both satisfy

$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2 \wedge \sigma.m1 = \sigma.m2$$

	$\alpha 1$	$\alpha 2$	m1	m2	q
$\sigma 1$	1	1	2	2	3
$\sigma 2$	101	101	102	102	103

Compose a state σ that is just like $\sigma 2$ except that it takes on the value of $\sigma 1$ for $\alpha 1$ and $\alpha 2$.

σ	1	1	102	102	103

σ also satisfies φ. $\alpha 1$ and $\alpha 2$ help satisfy φ independently of the values of other objects. The values of $\alpha 1$ and $\alpha 2$ taken from any state satisfying φ can be substituted for the values of $\alpha 1$ and $\alpha 2$ in $\sigma 2$; the resulting state will still satisfy φ. Whenever φ is A-autonomous, if $\sigma 1$ and $\sigma 2$ both satisfy φ, then $\sigma 2$ with $\sigma 1$ substituted at A will satisfy φ as well. Formally we define $\sigma 2$ with $\sigma 1$ substituted at A as

>> Def 5-3] $\sigma 2 \underset{A}{\frown} \sigma 1$

$$\sigma 2 \underset{A}{\frown} \sigma 1 \equiv_{def} \sigma \quad \text{where} \quad \sigma \underset{\bar{A}}{=} \sigma 2 \wedge \sigma.A = \sigma 1.A$$

Theorem 5-1]

φ is A-autonomous iff

$$(\forall \sigma 1, \sigma 2)(\varphi(\sigma 1) \wedge \varphi(\sigma 2) \supset. \varphi(\sigma 2 \underset{A}{\frown} \sigma 1))$$

The constraint

$$\varphi(\sigma) \quad \equiv \quad \sigma.\alpha 1 = \sigma.\alpha 2 \ \wedge \ \sigma.m1 = \sigma.m2$$

is $\{\alpha 1,\alpha 2\}$-autonomous. It is also $\{m1,m2\}$-autonomous. It is also q-autonomous for any arbitrary other object q. The value of q may change independently of any other object, especially since q is not constrained at all by φ. If we think of each relatively autonomous set of objects (e.g. $\{\alpha 1,\alpha 2\}$) as a single "pseudo-object", we can see that theorem 2-6

$$A \ \mathbb{D}^{H}_{\varphi} \beta \quad \supset \quad (\exists \alpha \in A)(\ \alpha \ \mathbb{D}^{H}_{\varphi} \beta \ )$$

generalizes to the following theorem.

Theorem 5-2]

If φ is A_i-autonomous, $i = 1,\ldots,k$ then

$$\cdot \ (\ \bigcup_{i=1}^{k} A_i \ ) \ \mathbb{D}^{H}_{\varphi} \beta \quad \supset \quad \bigvee_{i=1}^{k} \ (\ A_i \ \mathbb{D}^{H}_{\varphi} \beta \ )$$

If in some system constrained by the example φ above, information was transmitted from $\{\alpha 1,\alpha 2,m1,m2,q\}$ to β and β did not depend upon q or upon $\{m1,m2\}$, then β would certainly have to depend upon $\{\alpha 1,\alpha 2\}$.

If φ permits the value of each object to change independently of the value of any other object, then φ is α-autonomous for all α. This is the formal definition of autonomy (described informally in section 2.6).

>> Def 5-4] φ is autonomous iff

$$(\forall \alpha, \sigma 1, \sigma 2)(\ \varphi(\sigma 2) \ \wedge \ \varphi(\sigma 1) \ \supset . \ \varphi(\ \sigma 2 \underset{\alpha}{\sim} \sigma 1 \ )$$

----- Section 5.5 --- Strong Dependency Induction [relprf:]

Chapter 4 discussed Strong Dependency Induction for autonomous constraints only. The definitions and theorems in this section extend those results to non-autonomous constraints.

First we extend the definitions of section 2.3.

>> Def 5-5] $\sigma 1$ and $\sigma 2$ differ only at A and differ at B after H given φ

$$\sigma 1 \; {}^{\varphi}_{A}\Diamond^{H}_{B} \; \sigma 2 \;\equiv_{def}\; \sigma 1 \overset{\varphi}{\underset{A}{=}} \sigma 2 \;\wedge\; (\forall \beta \in B)\,(\, H(\sigma 1).\beta \neq H(\sigma 2).\beta \,)$$

>> Def 5-6] B strongly depends upon A after H given φ

$$A \; \mathbb{D}^{H}_{\varphi} \; B \;\equiv_{def}\; (\exists \sigma 1, \sigma 2)\,(\, \sigma 1 \; {}^{\varphi}_{A}\Diamond^{H}_{B} \; \sigma 2 \,)$$

>> Def 5-7] B strongly depends upon A given φ

$$A \; \mathbb{D}_{\varphi} \; B \;\equiv_{def}\; (\exists H)\,(\, A \; \mathbb{D}^{H}_{\varphi} \; B \,)$$

Theorem 5-3] (proof left to reader)

$$A \; \mathbb{D}^{H}_{\varphi} \; B \;\supset\; (\forall \beta \in B)\,(\, A \; \mathbb{D}^{H}_{\varphi} \; \beta \,)$$

We argued in section 4.2 that if information were transmitted from α to β by $\delta 1 \delta 2$, then there should be some intermediate object m such that $\delta 1$ transmits information from α to m and m transmits information from m to β. In the case of non-autonomous constraints, Strong Dependency may fail to mirror this intuition. Consider the system

$\delta 1$: (m1 ← α; m2 ← α)
$\delta 2$: β ← m1

initially constrained by the invariant but non-autonomous constraint

$$\varphi(\sigma) \;\equiv\; \sigma.m1 = \sigma.m2$$

Although we can directly show that $\alpha \; \mathbb{D}^{\delta 1 \delta 2}_{\varphi} \; \beta$, we find that

$$\neg\, m1 \; \mathbb{D}^{\delta 2}_{\varphi} \; \beta \qquad \text{as well as} \qquad \neg\, m2 \; \mathbb{D}^{\delta 2}_{\varphi} \; \beta$$

But, since φ is {m1,m2}-autonomous, we do find that

$$\{m1, m2\} \; \mathbb{D}^{\delta 2}_{\varphi} \; \beta$$

We also can show that α transmits information to both m1 and m2. That is

$$\alpha \; \mathbb{D}_{\varphi}^{\delta 1} \; \{m1, m2\}$$

In fact, generally we can show that

Theorem 5-4]

If φ is invariant then

$$A \; \mathbb{D}_{\varphi}^{HH'} \; \beta \supset. \quad (\exists M)\,(\, A \; \mathbb{D}_{\varphi}^{H} \; M \;\wedge\; M \; \mathbb{D}_{\varphi}^{H'} \; \beta \,)$$

This theorem, a generalization of theorem 4-1, follows immediately from the following theorem

Theorem 5-5]

If φ is invariant
and $M = \{\, m \mid H(\sigma 1).m \neq H(\sigma 2).m \,\}$ then

$$\sigma 1 \; {}_{A}\!\overset{\varphi}{\Diamond}_{\beta}^{HH'} \; \sigma 2 \quad \underline{\text{iff}} \quad \sigma 1 \; {}_{A}\!\overset{\varphi}{\Diamond}_{M}^{H} \; \sigma 2 \;\wedge\; H(\sigma 1) \; {}_{M}\!\overset{\varphi}{\Diamond}_{\beta}^{H'} \; H(\sigma 2)$$

Just as corollary 4-2 followed from theorem 4-1, we find that the following corollary follows from theorem 5-4

Corollary 5-6]

If φ is invariant and $\beta \notin A$ then

$$(\forall \delta, m)\,(\, A \; \mathbb{D}_{\varphi}^{\delta} \; m \;\supset\; m \in A \,) \;\vee\; (\forall \delta, M)\,(\, M \; \mathbb{D}_{\varphi}^{\delta} \; \beta \;\supset\; \beta \in M \,)$$

$$\supset. \quad \neg\, A \; \mathbb{D}_{\varphi} \; \beta$$

Chapter 6 - Non-invariant Constraints [noninv:]

----- Section 6.1 --- Introduction []

In sections 4.2 and 5.5 we explored Strong Dependency Induction for invariant constraints only. In this chapter, we will extend the inductive technique to include non-invariant constraints as well.

Induction using non-invariant constraints is useful when systems oscillate or pass through stages where one of a set of constraints is always satisfied. It is then possible to show the absence of information transmission by using Strong Dependency with respect to each of the constraints in the set separately. We call the set of constrainte a inductive cover.

We find that inductive covers are especially useful in analyzing sequential programs where they correspond to the inductive assertions attached to a program. Strong Dependency Induction can then be used to show absence of information transmission as the result of program execution.

----- Section 6.2 --- Constraint after a History [phist:]

As an initial constraint, φ characterizes the set of possible initial states of a system. In this section we show how to characterize the set of possible states after execution of some history.

If φ initially constrains a system, then after execution of history H, the set of possible states can be characterized as those states reachable by execution of H from a state satisfying φ initially. We write $[H]\varphi$ to characterize these states. Formally we define φ after H as

>> Def 6-1] $[H]\varphi$

$$[H]\varphi(\sigma') \quad =_{def} \quad \sigma' \in \{\, H(\sigma) \mid \varphi(\sigma) \,\}$$

If σ satisfies φ, then $H(\sigma)$ must satisfy $[H]\varphi$. Formally

Theorem 6-1] (proof left to reader)

$$\varphi(\sigma) \supset [H]\varphi(H(\sigma))$$

As an example, consider the system

$$\delta: \ \beta \leftarrow \alpha - 4$$
$$\varphi(\sigma) \equiv \sigma.\alpha < 10$$

We find that

$$[\delta]\varphi(\sigma) \equiv \sigma.c < 10 \ \wedge \ \sigma.\beta = \sigma.\alpha - 4$$

Execution of δ does not change α, so it remains less than 10. However, δ's execution guarantees that β will be $\alpha - 4$.

Note from the example above that $[H]\varphi$ need not be autonomous even if φ is. Note also that $[\delta]\varphi$ is stricter than φ. This increase in strictness occurs whenever φ is invariant.

Theorem 6-2] (proof left to reader)

If φ is invariant then

$$[H]\varphi \subseteq \varphi$$

----- Section 6.3 --- Strong Dependency Induction [noninduc:]

In using Strong Dependency Induction to determine whether information can be transmitted from A to β over execution of HH', we find some M such that information is transmitted from A to M over execution of H and from M to β over execution of H' (theorem 5-4). If the system is initially constrained by φ, then after execution of H, the system is constrained by $[H]\varphi$. To determine whether information can be transmitted from M to β over execution of H' after H has executed, one must consider a system constrained not by φ, but by $[H]\varphi$. Formally

Theorem 6-3]

$$A \; \mathbb{D}_{\varphi}^{HH'} \; \beta \supset (\exists M)\,(\, A \; \mathbb{D}_{\varphi}^{H} \; M \wedge M \; \mathbb{D}_{[H]\varphi}^{H'} \; \beta \,)$$

Note that the theorem holds even though $[H]\varphi$ need not be M-autonomous.

This theorem follows from the following theorem

Theorem 6-4] (proof similar to theorem 5-5)

If $M = \{\, m \mid H(\sigma 1).m \neq H(\sigma 2).m \,\}$ then

$$\sigma 1 \; {}_{A}\Diamond_{\beta}^{\varphi,HH'} \; \sigma 2 \quad \text{iff} \quad \sigma 1 \; {}_{A}\Diamond_{M}^{\varphi,H} \; \sigma 2 \wedge H(\sigma 1) \; {}_{M}\Diamond_{\beta}^{[H]\varphi,H'} \; H(\sigma 2)$$

If φ is invariant, then theorem 5-4 (the corresponding theorem for invariant φ) is seen to follow directly from theorems 6-3, 6-2 and 2-3.

The following corollary follows from theorem 6-3 as corollary 5-6 followed from theorem 5-4.

Corollary 6-5] (proof similar to theorem 5-6)

If $\beta \notin A$ then

$$(\forall H, \delta, m)\,(\, A \; \mathbb{D}_{[H]\varphi}^{\delta} \; m \;\supset.\; m \in A \,) \quad \vee$$

$$(\forall H, \delta, M)\,(\, M \; \mathbb{D}_{[H]\varphi}^{\delta} \; \beta \;\supset.\; \beta \in M \,)$$

$$\supset. \quad \neg \; A \; \mathbb{D}_{\varphi} \; \beta$$

If $[H]\varphi$ is autonomous for all H, the theorems in section 4.2 can be generalized as well. In particular we find that

Theorem 6-6] (proof similar to theorem 4-1)

If $(\forall H)\,(\, [H]\varphi$ is autonomous $)$ then

$$\alpha \; \mathbb{D}_{\varphi}^{HH'} \; \beta \supset (\exists m)\,(\, \alpha \; \mathbb{D}_{\varphi}^{H} \; m \wedge m \; \mathbb{D}_{[H]\varphi}^{H'} \; \beta \,)$$

----- Section 6.4 --- Inductive Covers [behcov:]

In this section, we explore Strong Dependency induction using inductive covers, sets of φi's, such that if some φ is true initially, one of the φi's will be true thereafter.

The simplest use of an inductive cover might be for an oscillating system. That is, $\varphi 1$ may be true initially, after execution of some operation, $\varphi 2$ will be true; after execution of another operation, $\varphi 1$ will be true again. We will present just such an example later in this section. More generally we define an inductive cover as a set of φi's, such that for every H, $[H]\varphi$ is contained in at least one of the φi's.

>> Def 6-2] $\{ \varphi i \}$ is an inductive cover for φ iff

$$(\forall H \exists i)\ (\ [H]\varphi \subseteq \varphi i\ )$$

Since each $[H]\varphi$ is contained in some φi, we find the following theorem follows directly from theorems 6-5 and 2-3.

Theorem 6-7]

If $\{ \varphi i \}$ is an inductive cover for φ then

$$(\forall \delta, m, i)\ (\ A \mathbin{\mathbb{D}^{\delta}_{\varphi i}} m \supset.\ m \in A\ ) \quad \vee$$

$$(\forall \delta, M, i)\ (\ M \mathbin{\mathbb{D}^{\delta}_{\varphi i}} \beta \supset.\ \beta \in M\ )$$

$$\supset.\quad \neg\ A \mathbin{\mathbb{D}_{\varphi}} \beta$$

A simple example of an oscillating system is

$$\delta:\ (\ \beta \leftarrow \alpha;\ \ \alpha \leftarrow -\alpha\ )$$

$$\varphi(\sigma) \equiv \sigma.\alpha = 37$$

It is easy to see that α is initially 37; after execution of δ, α will be -37; after execution of δ once more, α will be 37 again. No information can

be transmitted from α to β. α is constrained initially so that it contains no variety; there is none to convey. We will prove that $\neg\ \alpha \mathbb{D}_{\varphi} \beta$.

Instead of using the theorem above, we might first consider a retreat to the comfortable world of invariant constraints. φ is clearly not invariant. However, we could imagine finding an invariant $\varphi_{\star}$ containing φ such that

$$\neg\ \alpha \mathbb{D}_{\varphi_{\star}} \beta$$

By theorem 2-3, this would yield the desired result. Unfortunately, the most restrictive <u>invariant</u> $\varphi_{\star}$ containing φ is

$$\varphi_{\star}(\sigma) \equiv \sigma.\alpha = 37 \vee \sigma.\alpha = -37$$

This $\varphi_{\star}$ lets α exhibit some variety, that variety can be conveyed to β by execution of α, and therefore $\alpha \mathbb{D}_{\varphi_{\star}} \beta$, which is <u>not</u> the result desired.

We prove the desired result by using theorem 6-7, taking $\{\varphi 1, \varphi 2\}$ as an inductive cover for φ, where

$$\varphi 1(\sigma) \equiv \sigma.\alpha = 37$$
$$\varphi 2(\sigma) \equiv \sigma.\alpha = -37$$

Since both $\varphi 1$ and $\varphi 2$ eliminate all variety from α, we can show very easily that

$$M \mathbb{D}^{\delta}_{\varphi 1} \beta \supset \beta \in M \quad \text{and}$$

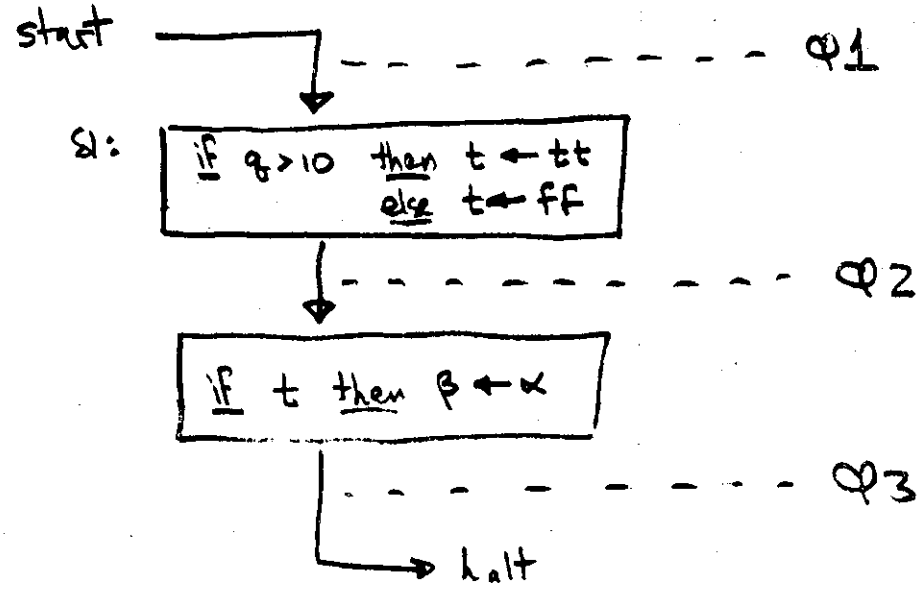
$$M \mathbb{D}^{\delta}_{\varphi 2} \beta \supset \beta \in M$$

So by theorem 6-7, we find that $\neg\ \alpha \mathbb{D}_{\varphi} \beta$.

----- Section 6.5 --- Information Transmission in Sequential Programs [infseq:]

In this section we will show how to prove the absence of information transmission in sequential programs by using Floyd assertions [Floyd 67] as an inductive cover.

Consider the flowchart program



Following [Lipton 73], this program can be modelled by the following computational system (pc acts as a program counter)

δ1: if pc = 1 then
(if q > 10 then t ← tt else t ← ff; pc ← 2)

δ2: if pc = 2 then
(if t then β ← α; pc ← 3)

constrained by φ which guarantees that execution begins at "start"

$$\varphi(\sigma) \equiv \sigma.pc = 1$$

Following [Floyd 67], we place an entry assertion at the beginning of the program, an exit assertion at the end, and intermediate assertions preceding each intermediate statement. Suppose that we know that the program only

executes on data that initially satisfies $\varphi 1$ (the entry assertion). Now let $\varphi 2,\ldots,\varphi n$ be assertions placed preceding statements labelled $\delta 2,\ldots,\delta n$ respectively, and let φ_{n+1} be the exit assertion (see diagram above).

The meaning of Floyd assertions is this: if the entry assertion ($\varphi 1$) is satisfied, and if control is at δi (i.e. $\sigma.pc = i$), then φi is true. Initially, control is at statement $\delta 1$, and if the entry assertion is satisfied, the state of the system can be characterized by $\varphi 1 \wedge \varphi$.

Control is always at some δi, therefore, some φi must <u>always</u> be true. That is just the requirement that makes { φi } an inductive cover for $\varphi 1 \wedge \varphi$.

It is useful to take the pc explicitly into account. Define

$$\varphi i\!\!\uparrow(\sigma) \equiv \varphi i(\sigma) \wedge \sigma.pc = i$$

Since the value of the pc is i whenever control is at δi, $\varphi i\!\!\uparrow$ is always true when control is at δi, and therefore { $\varphi i\!\!\uparrow$ } is also an inductive cover for $\varphi 1 \wedge \varphi$ ($\varphi 1\!\!\uparrow$).

Now we see that

$$(\forall \sigma)(\ \varphi i\!\!\uparrow(\sigma) \supset \sigma.pc = i\ )$$

and each δj is of the form

δj: <u>if</u> pc = j <u>then</u> ...

so if $x \mathbb{D}^{\delta j}_{\varphi i\uparrow} y$, then (unless $y \in x$ - see section 2.5) i must be equal to j, for otherwise execution of δj can have no effect on y (or any object). Formally

$$(\forall i,j,X,y)(\ X \mathbb{D}^{\delta j}_{\varphi i\uparrow} y \supset .\ i = j \vee y \in X\ )$$

Thus by theorem 6-7, to show $\neg A \mathbb{D}_{\varphi} \beta$, we need only show that

1. Either $(\forall i,m)(A \mathbb{D}^{\delta i}_{\varphi i\star} m \supset m \in A)$

2. Or $(\forall i,M)(M \mathbb{D}^{\delta i}_{\varphi i\star} \beta \supset \beta \in M)$

The second alternative corresponds to the following proof technique for showing that no information can be transmitted from α to β.

> For each statement δ_i that contains an assignment to β, show that $\varphi i\star$ constrains the state so that no information can be transmitted to β as a result of execution of δ_i. $\varphi i\star$ is the inductive assertion for statement δ_i conjoined with $\sigma.pc = i$. [We need not be concerned with statements that cannot assign to β; they can never transmit information to β.]

In the example above, we pick the entry assertion to be

$$\varphi 1(\sigma) \equiv \sigma.q < 10$$

We can then show that $\varphi 2$ is a legal inductive assertion for statement $\delta 2$

$$\varphi 2(\sigma) \equiv \neg\sigma.t$$

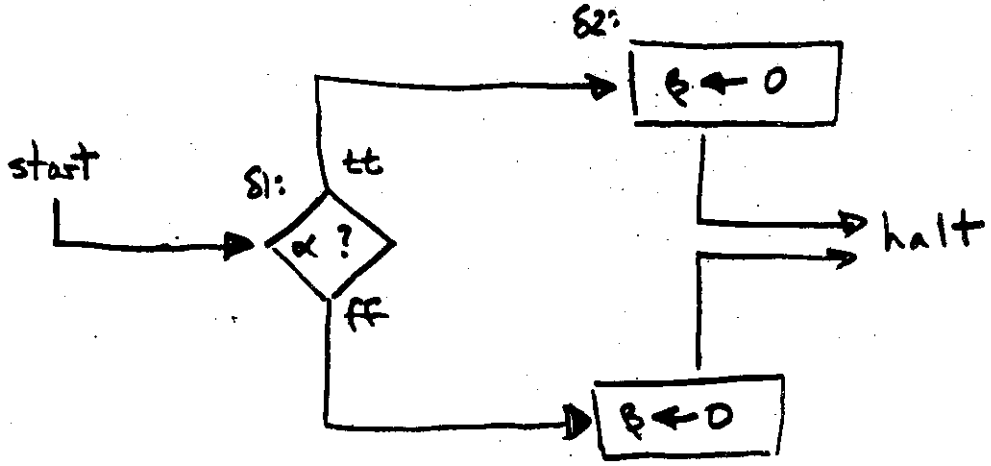
Since q is initially less than 10, t must be false when control reaches $\delta 2$ (by execution of $\delta 1$). Since t is false, execution of $\delta 2$ can never transmit information to β. Formally,

$$M \mathbb{D}^{\delta 2}_{\varphi 2\star} \beta \supset \beta \in M$$

Since $\delta 2$ is the only statement that assigns to β, we have shown that no information can be transmitted from α at β over execution of the program. In general, suppose that β is <u>only</u> assigned to at statement k (not necessarily the last statement). Then, in order to prove $\neg A \mathbb{D}_{\varphi} \beta$, we need <u>only</u> show

$$(\forall M)(M \mathbb{D}^{\delta k}_{\varphi k\star} \beta \supset \beta \in M)$$

Yet, there are difficulties in using Strong Dependency as a model of information transmission in programs. Consider the flowchart



which can be modelled by the constrained system

δ1: if pc = 1 then (if α then pc ← 2 else pc ← 3)
δ2: if pc = 2 then (β ← 0; pc ← 4)
δ3: if pc = 3 then (β ← 0; pc ← 4)
$\Phi(\sigma) \equiv \sigma.pc = 1$

Now it is clear from looking at the program that information cannot be transmitted from α to β, since β is set to 0 regardless of α's value. Yet we find that

$$\alpha \, \triangleright_{\Phi}^{\delta 1 \delta 2} \, \beta \qquad \text{and therefore} \qquad \alpha \, \triangleright_{\Phi} \, \beta$$

This can be demonstrated by

Picking σ1 so that $\sigma 1.\alpha = tt$, $\sigma 1.\beta = 37$
Picking σ2 just like σ1 except that $\sigma 2.\alpha = ff$

Then $\sigma 1 \underset{\alpha}{\overset{\Phi}{=}} \sigma 2$, $(\delta 1 \delta 2)(\sigma 1).\beta = 0$, $(\delta 1 \delta 2)(\sigma 2).\beta = 37$

This example may appear to invalidate the Strong Dependency Hypothesis. In fact, it does not. The Strong Dependency formalism implicitly assumes that β's observer knows the history being executed. Suppose that an observer of β did know that δ1δ2 was being executed. δ2 has an effect only if the pc is 2. If δ2 does have an effect on β, then β's observer can infer that the pc was 2 when δ2 was executed, which implies that α was true initially. That information about α is thus transmitted to β.

In arguing that information cannot be transmitted from α to β, we tacitly made the assumption that β's observer could not observe the history executed. Ordinarily, we might instead make the assumption that β's observer can only detect the passage of time (as well as the value of β of course). Work in progress (section 7.3) formalizes the observation of time and allows us to show formally that, in the example above, as long as only time, and not the history, can be observed, no information can be transmitted from α to β.

Chapter 7 - Work in Progress [infwrk:]

----- Section 7.1 --- Introduction []

In this chapter, we discuss work in progress, both extensions to the Strong Dependency Model, as well as other models suggested by issues raised in exploring Strong Dependency.

----- Section 7.2 --- Alternate Models for Information Transmission [infalt:]

We have found that Strong Dependency corresponds to information transmission only in autonomously constrained systems. For example, in the system

$$\delta: \quad \beta \leftarrow \alpha 1$$
$$\varphi(\sigma) \equiv \sigma.\alpha 1 = \sigma.\alpha 2$$

information can certainly be transmitted from $\alpha 1$ to β, yet we find that $\neg\ \alpha 1 \not\rhd_{\varphi} \beta$.

Two other models, Inferential Dependency and Direct Dependency, are being explored in an attempt to extend Strong Dependency to non-autonomous constraints. The two models treat "inferential" transmission differently. Inferential Dependency would indicate that information is transmitted from both $\alpha 1$ and $\alpha 2$ to β in the example above. Direct Dependency would indicate only that information is transmitted from $\alpha 1$ to β. The advantage of a Direct Dependency formalism can be seen more clearly in the following example:

$$\delta: \quad \beta \leftarrow \alpha 1$$
$$\varphi(\sigma) \equiv \sigma.\alpha 1.tag = \sigma.\alpha 2.tag$$

Information is certainly transmitted from $\alpha 1$ to β by execution of δ. Since φ indicates that the tag component of $\alpha 1$ and $\alpha 2$ are the same, one might well conclude that some information about $\alpha 2$ is transmitted to β as well. If the

tag component does not contain important information (i.e. we don't care if it is transmitted), we may find it useful to ignore this inferential transmission. A Direct Dependency formalism would do just that.

If a model of information transmission <u>does</u> include the effect of "inferential" transmission, information transmission cannot be monotonic in the sense of theorem 2-3. More restrictive constraints might <u>increase</u> the sources of information. For example in the system described above, Φ is more restrictive than the always true constraint (i.e. no constraint at all), yet imposing Φ <u>adds</u> an information path (from $\alpha 2$ to β).

The Inferential Dependency formalism is is being developed from a purely inferential, rather than an information theoretic approach. We say that β inferentially depends upon α after execution of H in a system constrained by Φ, if an observer of the system, able to view only β can make some <u>inference</u> about α that "says more" about α than can be determined from Φ alone. We find that the definition of "says more" is the crucial (and most interesting) part of this model.

Our investigations to date indicate that the model is at least as general as Strong Dependency, in the sense that we can show that Inferential Dependency and Strong Dependency give the same results for relatively-autonomous constraints.

The definition of "says more" turns out to be related to what can be called "contingent" information transmission. In execution of

$$\delta: \quad \beta \leftarrow ((\alpha 1 + \alpha 2) \bmod 128)$$

information is clearly transmitted from $\{\alpha 1, \alpha 2\}$ to β. It is not so clear that information is transmitted from $\alpha 1$ alone to β. No matter what an observer finds to be the value of β after execution of δ, no inference can be made about the value of $\alpha 1$. $\alpha 1$ can take on any value <u>contingent</u> on the value of $\alpha 2$. Strong Dependency would indicate that β does depend upon $\alpha 1$. We find that we can define Inferential Dependency in two different ways; one would indicate contingent information transmission, one would not.

Theorem 2-1 holds precisely because Strong Dependency does indicate

contingent information transmission. In a model that ignores contingent information transmission, information might be transmitted from a set of objects A to β, even though no information might be transmitted to β from any one single object in A.

One probable prerequisite for any acceptable model of information transmission is an induction principle at least as general as Strong Dependency Induction (theorems 5-4 and 6-3) and a theorem that permits separation of variety in a manner analogous to theorem 4-5.

----- Section 7.3 --- Mechanisms [infmech:]

In this paper, we have assumed that information problems may only be solved by imposing an initial constraint on a system. As we note in [Cohen 76], problems may also be solved by adding a mechanism to a system. In [Cohen 76], we define a mechanism as implementing an arbitrary mapping from an augmented system (as it is provided to a user) to an original base system. This mechanism formalism can be used to model protection mechanisms, synchronization mechanisms, sequential and concurrent control mechanisms, virtual machine monitors, and can be used to model information hiding and situations in which a user is to be prevented from observing the exact sequence of operations performed in the base system in response to execution of operations executed by the user in the augmented system.

[Rotenberg 73] and [Denning 75] have warned us that we must be careful in adding mechanisms to a system. For even as the mechanisms may eliminate certain information paths, they may covertly add others. [Rotenberg 73] especially provides a number of exceedingly subtle examples of covert information paths. Our formal model of mechanism, in conjunction with the Strong Dependency formalism, permits a characterization of those mechanisms that do not add new paths for information transmission.

Two sorts of run-time mechanisms that prevent information transmission have appeared in the literature. The *-property mechanism [Bell & LaPadula 73] requires that the classification of ordinary objects (not processes) be fixed. [Denning 75] has shown that such mechanisms do prevent information transmission without adding covert channels.

If the classification of objects are allowed to vary depending upon the information stored in them, then covert information paths are easily introduced. The Adept-50 system [Weissman 69] does allow the classification of objects to vary; [Denning 76] has shown that it permits covert leakage of information. We are exploring mechanisms that permit classifications to vary; we can prove that covert information paths are not introduced because we also require that the state of the system be initially constrained. The initial constraints, correspond to initial properties of an access matrix.

A mechanism may be used as a formal tool for specifying the mapping from a given system to a simpler system that may be easier to analyze. Work in progress examines classes of mechanisms that preserve various information transmission properties.

Finally, we noted above that the mechanism formalism is useful for specifying exactly which parts of the behavior of a system can be observed. In section 6.5, we noted that if only the time of a computation can be observed instead of the history, certain information paths disappear. We can formalize this argument through the use of those mechanisms called "sequential control mechanisms" in [Cohen 76].

----- Section 7.4 --- Information Theory [infthr:]

Strong Dependency and the other models of information transmission alluded to above are non-quantitative. They indicate whether information can be transmitted, but not how much. A number of different measures can be formulated, depending upon one's approach to contingent and inferential information transmission. Each of these measures may be based on Shannon's information entropy [Shannon & Weaver 49].

The following example illustrates the reason for two different measures corresponding to two different approaches to contingent information transmission (section 7.2).

δ: $\beta \leftarrow ((\alpha 1 + \alpha 2) \bmod 128)$

If initially, $\alpha 1$ and $\alpha 2$ can take on values from 0 to 127 with equal

probability, then execution of δ transmits 7 bits of information from {α1,α2} to β. But how many bits of information are transmitted to β from α1 alone?

The answer might reasonably be zero, for reasons identical to those given in section 7.2. An observer of β can gain no information about the value of α1 alone. In information theoretic terms we might say that the equivocation of β with respect to α1 is 7 bits; any value of β observed estimates any initial value of α1 with 7 bits worth of uncertainty. Since α1 has an initial entropy of 7 bits (the values 0 to 127 can initially occur with equal probability), the amount of information transmitted is 7 - 7 (initial entropy - equivocation) or zero bits.

One might instead measure the average number of bits transmitted from α1 to β, averaging over all the possible ways in which each object but α1 is held constant. If α2 is held constant, then the full variety of α1 (7 bits worth) is transmitted to β by execution of δ. The average number of bits (averaged over the values of α2) transmitted from α1 to β is 7.

A quantitative model of information transmission might also include the effect of constraint. Constraint reduces the variety in a system. We might write b(A-(Φ::H)->β) to mean the number of bits of information transmitted from A to β in a system constrained by Φ over execution of H. Increasing the constraint in a system reduces the variety available to be conveyed. We might expect an appropriate definition for b to be monotonic.

$$\Phi 1 \subseteq \Phi 2 \supset. \quad b(A-(\Phi 1::H)->\beta) \leq b(A-(\Phi 2::H)->\beta)$$

although due to the effects of inference (section 7.2), this relationship should perhaps only hold for A-autonomous constraints.

We ask the question - is it desirable or useful, and if so, then possible to define b so that

$$b(A1-(\Phi::H)->\beta) + b(A2-(\Phi::H)->\beta) = b((A1 \cup A2)-(\Phi::H)->\beta)$$

Neither of the alternatives suggested above satisfy this additive property. We might argue that if A1 transmits v1 bits to β and A2 transmits v2 bits to

β, one might think that A1 ∪ A2 transmits v1 + v2 bits to β. If b is not defined in a such a way as to satisfy this property, then the difference between the left and right hand sides of the equation might be construed as measuring the relative interference between A1 and A2 in transmitting information to β over execution of H.

We have implicitly assumed above that each state satisfying Φ occurs with equal probability. More generally, the actual number of bits transmitted over some history must depend upon the distribution, pr, of the initial states. In this sense, pr is a generalization of an initial constraint Φ. We might write $b(A\text{-}(pr::H)\text{->}\beta)$ to mean the number of bits of information transmitted from A to β as a result of execution of H.

If $pr(\sigma)$ is the probability that σ is an initial state, then one can define $[H]pr$ so that $([H]pr)(\sigma)$ is the probability of state σ occuring after execution of H. One might expect a quantitative theory of information to satisfy the following property corresponding roughly to Strong Dependency Induction.

If $b(A\text{-}(pr::HH')\text{->}\beta) = k$ then

There exists some set of objects M such that
$b(A\text{-}(pr::H)\text{->}M) \geq k$
$b(M\text{-}([H]pr::H')\text{->}\beta) \geq k$

where $b(X\text{-}(pr::H)\text{->}Y) \equiv_{def} \sum_{y \in Y} b(X\text{-}(pr::H)\text{->}y)$

That is, if execution of HH' transmits k bits of information from A to β, there must be some set of objects M, so that execution of H transmits at least k bits from A to M and subsequent execution of H' transmits at least k bits from M to β.

----- Section 7.5 --- Declassification [confnm:]

Throughout this paper, we have considered those problems where we want to guarantee that information is not transmitted from one set of objects to another set of objects. These problems do not take into consideration the matter of declassification. [Bell & LaPadula 73] have extended their *-property mechanism to permit trustworthy executors to transmit information where such transmission would not normally be permitted. Similarly, we need to extend our notion of information problem to formally model declassification by trustworthy executors.

We are currently exploring a definition of the Confinement Problem that does formally model such declassification. We expect to show that access matrix systems of the form suggested in [Cohen & Jefferson 75] can indeed be used to solve just that problem.

Chapter 8 - Conclusion [iconcl:]

This paper has introduced Strong Dependency, a formalism for describing information transmission in computational systems. We showed how the formalism could be used to describe information problems and prove the correctness of solutions to them.

The notation $A \Rightarrow \beta$ means that β strongly depends on A. That is, over execution of some history H, some change in the initial values of the objects in A may cause a corresponding change in the value of β; variety in A can be conveyed to β. We argued in this paper that $A \Rightarrow \beta$ corresponds to the intuitive notion that information is transmitted from the set of objects A to β.

We found that by imposing some initial constraint on the system, the variety in an object could be reduced, thereby preventing information transmission. $A \Rightarrow_{\varphi} \beta$, β strongly depends on A given φ, corresponds to the intuitive notion that information can be transmitted from A to β in a system constrained by φ as long as φ is autonomous relative to A, that is, as long as φ does not establish some correspondence between the values of objects in A and those not in A.

We define a solution to an information problem as an initial constraint φ that will prevent certain specified information transmission. For example, the problem of guaranteeing that no information can be transmitted from α to β can be written as

$$\chi(\varphi) \equiv \neg\, \alpha \Rightarrow_{\varphi} \beta$$

We say that φ solves χ if φ prevents information transmission from α to β.

As Strong Dependency is defined, it is necessary to show that no information can be transmitted from A to β over <u>every</u> possible history in order to show that no information can be transmitted from A to β.

We therefore introduced Strong Dependency Induction, an inductive technique for proving correctness of solutions to information problems.

Strong Dependency Induction is based on the principle that when information is transmitted from α to β over execution of HH', there is some intermediate object m, such that execution of H transmits information from α to m and execution of H' transmits information from m to β.

We found that Strong Dependency Induction is ineffective if the Strong Dependency relation is not transitive. We introduced another proof technique, Separation of Variety, that may be used in conjunction with Strong Dependency Induction in case Strong Dependency is non-transitive.

We discussed Strong Dependency (and Strong Dependency Induction) first for constraints both autonomous (those constraining the variety in an object independently of other objects) and invariant, extending the results to relatively-autonomous and non-invariant constraints respectively.

Finally we noted that in a computational system modelling execution of a sequential program, the initial constraint Φ corresponds to an entry assertion for the program. The set of inductive assertions attached to the program can be used in conjunction with the Strong Dependency formalism to show absence of information transmission as a result of program execution for any input satisfying the entry assertion.

Strong Dependency is a first approximation to an understanding of information transmission in computational systems. The chapter detailing work in progress represents a collection of the directions for future research.

ACKNOWLEDGEMENTS

It is a pleasure to thank Anita Jones. Her ideas have impacted this paper in a variety of ways. Paul Hilfinger and Jack Mostow have helped me debug a number of models of information transmission. Eric Ostrom pointed out the relevance of information theory to this research. Bill Wulf, Dorothy Denning and Bruce Lindsay provided useful comments on an earlier draft of this paper. Doug Clark, John Gaschnig, Gary Goodman and Mike Shamos acted as critical sounding boards for a number of the ideas presented here.

Appendix A - Proofs

Theorem 2-1

see theorem 2-6 with $\varphi \equiv tt$

Theorem 2-4

Given

1] $(\forall \alpha \in A)(\neg\ A \ \mathbb{D}^{\lambda}_{\varphi}\ \alpha)$

Prove: $(\forall \beta)(\neg\ A \ \mathbb{D}_{\varphi}\ \beta)$

2] Assume $\sigma 1 \overset{\varphi}{\underset{A}{=}} \sigma 2$

3] $(\forall \alpha \in A)(\sigma 1.\alpha = \sigma 2.\alpha)$ [1,2]
4] $\sigma 1.A = \sigma 2.A$ [3]
5] $\sigma 1 = \sigma 2$ [2,4]
6] $H(\sigma 1).\beta = H(\sigma 2).\beta$ [5]
7] $\neg\ A \ \mathbb{D}_{\varphi}\ \beta$ [2-6]

Theorem 2-5

Given

1] $\beta \notin A$

Prove: $\neg\ A \ \mathbb{D}^{\lambda}_{\varphi}\ \beta$

2] Assume $\sigma 1 \overset{\varphi}{\underset{A}{=}} \sigma 2$

3] $\sigma 1.\beta = \sigma 2.\beta$ [1,2]

4] $\neg\ A \ \mathbb{D}^{\lambda}_{\varphi}\ \beta$ [2-3]

Strong Dependency (A)

Theorem 2-6

Given:

1] φ is autonomous

2] $A \; \mathbb{D}^{H}_{\varphi} \; \beta$

Prove: $(\exists \alpha \subset A)(\alpha \; \mathbb{D}^{H}_{\varphi} \; \alpha)$

3] $(\forall \alpha \subset A)(\varphi$ is α-autonomous) [1, def 5-4, th 5-1]

4] $(\bigcup_{\alpha \subset A} \alpha) \; \mathbb{D}^{H}_{\varphi} \; \beta$ [2]

5] $\bigvee_{\alpha \subset A} (\alpha \; \mathbb{D}^{H}_{\varphi} \; \beta)$ [3,4, th 5-2]

Theorem 3-1

Given

1] $\chi(\varphi) \equiv \neg A \; \mathbb{D}_{\varphi} \; \beta \wedge \varphi$ is A-independent

2] $\chi(\varphi 1) \wedge \chi(\varphi 2)$

Prove: $\chi(\varphi 1 \vee \varphi 2)$

3] $\varphi 1$ is A-independent [1,2]
4] $\varphi 2$ is A-independent [1,2]
5] $\varphi 1 \vee \varphi 2$ is A-independent [3,4, left to reader]
6] Assume $\sigma 1 \overset{\varphi 1 \vee \varphi 2}{\underset{A}{=}} \sigma 2$

7] $\sigma 1 \underset{A}{=} \sigma 2$ [6]

8] $\varphi(\sigma 1) \vee \varphi(\sigma 2)$ [6]
9] Case 1 $\varphi 1(\sigma 1)$
10] $\varphi 1(\sigma 2)$ [9,7,3]

11] $\sigma 1 \overset{\varphi 1}{\underset{A}{=}} \sigma 2$ [9,10,7]

12] $\neg$ A $\mathbb{D}^{H}_{\varphi 1}$ β [1,2]

13] $H(\sigma 1).\beta = H(\sigma 2).\beta$ [11,12]
14] Case 2 $\varphi 2(\sigma 1)$
15] $H(\sigma 1).\beta = H(\sigma 2).\beta$ [Similar to 9-13]
16] $H(\sigma 1).\beta = H(\sigma 2).\beta$ [8,9-13,14-15]

17] $\neg$ A $\mathbb{D}^{H}_{\varphi 1 \vee \varphi 2}$ [6-16]

18] $X(\varphi 1 \vee \varphi 2)$ [17,5,1]

Theorem 4-1

Given

1] φ is autonomous
2] φ is invariant

3] α $\mathbb{D}^{HH'}_{\varphi}$ β

Prove: $(\exists m)(\ \alpha\ \mathbb{D}^{H}_{\varphi}\ m\ \wedge\ m\ \mathbb{D}^{H'}_{\varphi}\ \beta\ )$

4] $(\exists M)\,(\,\alpha \; \mathbb{D}_{\varphi}^{H} M \;\wedge\; M \; \mathbb{D}_{\varphi}^{H'} \beta\,)$ [2,3, th 5-4]

5] $(\forall m \in M)\,(\,\alpha \; \mathbb{D}_{\varphi}^{H} m\,)$ [4, th 5-3]

6] $(\exists m \in M)\,(\,m \; \mathbb{D}_{\varphi}^{H'} \beta\,)$ [1,4, th 2-6]

Q.E.D. [5,6]

Theorem 4-2

1] φ is autonomous
2] φ is invariant
3] $\beta \neq_{H} \alpha$
4] $\alpha \; \mathbb{D}_{\varphi}^{H} \beta$

Prove: $(\exists \delta, m \neq \alpha)\,(\,\alpha \; \mathbb{D}_{\varphi}^{\delta} m\,) \;\wedge\; (\exists \delta, m \neq \beta)\,(\,m \; \mathbb{D}_{\varphi}^{\delta} \beta\,)$

5] $(\exists \delta, m \neq \alpha)\,(\,\alpha \; \mathbb{D}_{\varphi}^{\delta} m\,)$ [2,3,4, th 5-6]

6] $(\exists \delta, M)\,(\,M \; \mathbb{D}_{\varphi}^{\delta} \beta \;\wedge\; \beta \notin M\,)$ [2,3,4, th 5-6]

7] $(\exists \delta, m \neq \beta)\,(\,m \; \mathbb{D}_{\varphi}^{\delta} \beta\,)$ [6,1, th 2-6]

Q.E.D . [5,7]

Theorem 4-3

Given

1] φ is autonomous
2] φ is invariant
3] q is reflexive
4] q is transitive
5] $(\forall x, y, \delta)\,(\,x \; \mathbb{D}_{\varphi}^{\delta} y \;\supset\; q(x, y,)\,)$

Prove: $(\forall x, y, H)\,(\,x \; \mathbb{D}_{\varphi}^{H} y \;\supset\; q(x, y)\,)$
by induction on length of H
Base $H = \lambda$. Follows from [3, th 2-5]
Induction Assume for H, prove for Hδ

6] Assume $x \; \mathbb{D}_{\varphi}^{H\delta} \; y$

7] $(\exists m)(\alpha \; \mathbb{D}_{\varphi}^{H} \; m \wedge m \; \mathbb{D}_{\varphi}^{\delta} \; y)$ [1,2,6, th 4-1]
8] $q(x,m)$ [7, induction]
9] $q(m,y)$ [7,5]
10] $q(x,y)$ [8,9,4]

Theorem 4-5

Given

1] $\{ \varphi_i \}$ is an A-independent cover

2] $A \; \mathbb{D}_{\varphi}^{H} \; \beta$

Prove: $(\exists i)(A \; \mathbb{D}_{\varphi \wedge \varphi_i}^{H} \; \beta)$

3] $(\exists \sigma 1, \sigma 2)(\sigma 1 \underset{A}{\overset{\varphi}{=}} \sigma 2 \wedge H(\sigma 1).\beta \neq H(\sigma 2).\beta)$ [2]
4] $(\exists i)(\varphi_i(\sigma 1))$ [1]
5] $(\forall j)(\varphi_j$ is A-indpendent) [1]
6] $\sigma 1 \underset{A}{=} \sigma 2$ [3]

7] $(\exists i)(\varphi_i(\sigma 1) \wedge \varphi_i(\sigma 2))$ [4,5,6]

8] $(\exists i)(\sigma 1 \underset{A}{\overset{\varphi \wedge \varphi_i}{=}} \sigma 2 \wedge H(\sigma 1).\beta \neq H(\sigma 2).\beta)$ [3,7]

9] $(\exists i)(A \; \mathbb{D}_{\varphi \wedge \varphi_i}^{H} \; \beta)$ [8]

Theorem 5-1

Prove: φ is A-autonomous iff
$(\forall \sigma 1, \sigma 2)(\varphi(\sigma 1) \wedge \varphi(\sigma 2) \supset . \varphi(\sigma 2 \underset{\sim}{/}_{A} \sigma 1))$

1] ==> Assume φ is A-autonomous
2] $\varphi = \varphi 1 \wedge \varphi 2$, $\varphi 1$ is A-strict, $\varphi 2$ is A-independent [1]
3] Assume $\varphi(\sigma 1) \wedge \varphi(\sigma 2)$
4] $(\sigma 2 \underset{\sim}{/}_{A} \sigma 1).A = \sigma 1.A$ [Def 5-3]

5] $\varphi 1(\sigma 1)$ [2,3]
6] $\varphi 1(\sigma 2 \underset{\sim}{/}_{A} \sigma 1)$ [4,5,2, def 5-1]

7] $(\sigma 2 \underset{\sim}{/}_{A} \sigma 1) \underset{=}{}_{A} \sigma 2$ [Def 5-3]

8] $\varphi 2(\sigma 2)$ [2,3]
9] $\varphi 2(\sigma 2 \underset{\sim}{/}_{A} \sigma 1)$ [7,8,2, def 3-1]

10] $\varphi(\sigma 1) \wedge \varphi(\sigma 2) \supset . \varphi(\sigma 2 \underset{\sim}{/}_{A} \sigma 1)$ [3-(2,6,9)]

11] <== Assume $(\forall\sigma 1,\sigma 2)(\ \varphi(\sigma 1) \wedge \varphi(\sigma 2) \supset .\ \varphi(\ \sigma 2 \underset{A}{\sim} \sigma 1\ )\ )$

12] Let $\varphi 1(\sigma) \equiv_{def} (\exists\sigma\star)(\ \sigma\star \underset{A}{=} \sigma \wedge \varphi(\sigma\star)\ )$

13] Assume $\sigma 1 \underset{A}{=} \sigma 2$

14] $\varphi 1(\sigma 1) \equiv \varphi 1(\sigma 2)$ [12,13]

15] $\varphi 1$ is A-independent [13-14, def 3-1]

16] Let $\varphi 2(\sigma) \equiv_{def} (\exists\sigma\star)(\ \sigma\star.A = \sigma.A \wedge \varphi(\sigma\star)\ )$

17] Assume $\sigma 1.A = \sigma 2.A$

18] $\varphi 2(\sigma 1) \equiv \varphi 2(\sigma 2)$ [16,17]

19] $\varphi 2$ is A-strict [17-18, def 5-1]

20] Assume $\varphi(\sigma)$

21] $(\exists\sigma\star)(\ \sigma\star \underset{A}{=} \sigma \wedge \varphi(\sigma\star)\ )$ [20, $\sigma\star = \sigma$]

22] $(\exists\sigma\star)(\ \sigma\star.A = \sigma.A \wedge \varphi(\sigma\star)\ )$ [20, $\sigma\star = \sigma$]

23] $\varphi 1(\sigma) \wedge \varphi 2(\sigma)$ [12,16,21,22]

24] $\varphi(\sigma) \supset .\ \varphi 1(\sigma) \wedge \varphi 2(\sigma)$ [20-23]

25] Assume $\varphi 1(\sigma) \wedge \varphi 2(\sigma)$

26] $\sigma 1\star \underset{A}{=} \sigma \wedge \varphi(\sigma 1\star)$ [25,12]

27] $\sigma 2\star.A = \sigma.A \wedge \varphi(\sigma 2\star)$ [25,16]

28] $\varphi(\ \sigma 1\star \underset{A}{\sim} \sigma 2\star\ )$ [26,27,11]

29] $\sigma = \sigma 1\star \underset{A}{\sim} \sigma 2\star$ [26,27]

30] $\varphi 1(\sigma) \wedge \varphi 2(\sigma) \supset .\ \varphi(\sigma)$ [25-(28,29)]

31] φ is A-autonomous [15,19,24,30]

Theorem 5-2

Prove: If φ is A_i-autonomous, $i = 1,\ldots,k$

then $(\ \bigcup_{i=1}^{k} A_i\ )\ \mathbb{D}_{\varphi}^{H}\beta \supset \bigvee_{i=1}^{k} (\ A_i\ \mathbb{D}_{\varphi}^{H}\beta\ )$

by induction on k

Base k = 1. Direct by substitution

Induction Assume for k, prove for k+1

Strong Dependency (A)

1] Assume φ is A_i-autonomous, $i = 1,\ldots,k+1$

2] Let $A = A_{k+1}$, $A\!\!\star = \bigcup_{i=1}^{k} A_i$

3] Assume $(A\!\!\star \cup A) \not\mathbb{D}_{\varphi}^{H} \beta$

4] $\sigma 1 \overset{\varphi}{=}_{A\!\!\star \cup A} \sigma 2 \;\wedge\; H(\sigma 1).\beta \neq H(\sigma 2).\beta$ [3]

5] Let $\sigma = \sigma 1 \;\widetilde{A}\; \sigma 2$

6] $\varphi(\sigma)$ [1,2,4,5, th 5-1]

7] Case 1 $H(\sigma).\beta \neq H(\sigma 1).\beta$

8] $\sigma \overset{\varphi}{=}_{A} \sigma 1$ [4,5,6]

9] $A_{k+1} \not\mathbb{D}_{\varphi}^{H} \beta$ [7,8,2]

10] Case 2 $H(\sigma).\beta = H(\sigma 1).\beta$

11] $H(\sigma).\beta \neq H(\sigma 2).\beta$ [4,10]

12] $\sigma \overset{\varphi}{=}_{A\!\!\star} \sigma 2$ [4,5,6]

13] $A\!\!\star \not\mathbb{D}_{\varphi}^{H} \beta$ [11,12]

14] $\bigvee_{i=1}^{k} (A_i \not\mathbb{D}_{\varphi}^{H} \beta)$ [13,1,2, induction]

15] $\bigvee_{i=1}^{k+1} (A_i \not\mathbb{D}_{\varphi}^{H} \beta)$ [7-9,10-14]

Theorem 5-5

Given

1] φ is invariant

2] $M = \{ m \mid H(\sigma 1).m \neq H(\sigma 2).m \}$

Prove: $\sigma 1 \; {}^{\varphi}_{A}\!\Diamond^{HH'}_{\beta} \; \sigma 2$ iff $\sigma 1 \; {}^{\varphi}_{A}\!\Diamond^{H}_{M} \; \sigma 2 \;\wedge\; H(\sigma 1) \; {}^{\varphi}_{M}\!\Diamond^{H'}_{\beta} \; H(\sigma 2)$

Strong Dependency (A)

3] ==> Assume $\sigma 1 \; {}^{\varphi}_{A}\Diamond^{HH'}_{\beta} \; \sigma 2$

4] $\sigma 1 \overset{\varphi}{\underset{A}{=}} \sigma 2$ [3]

5] $(\forall m \in M)\,(\, H(\sigma 1).m \neq H(\sigma 2).m \,)$ [2]

6] $\sigma 1 \; {}^{\varphi}_{A}\Diamond^{H}_{M} \; \sigma 2$ [4,5]

7] $\varphi(H(\sigma 1)) \wedge \varphi(H(\sigma 2))$ [1,4]

8] $H(\sigma 1) \underset{M}{=} H(\sigma 2)$ [2]

9] $H'(H(\sigma 1)).\beta \neq H'(H(\sigma 2)).\beta$ [3]

10] $H(\sigma 1) \; {}^{\varphi}_{M}\Diamond^{H'}_{\beta} \; H(\sigma 2)$ [7,8,9]

11] <== Assume $\sigma 1 \; {}^{\varphi}_{A}\Diamond^{H}_{M} \; \sigma 2 \wedge H(\sigma 1) \; {}^{\varphi}_{M}\Diamond^{H'}_{\beta} \; H(\sigma 2)$

12] $\sigma 1 \overset{\varphi}{\underset{A}{=}} \sigma 2$ [11]

13] $(HH')(\sigma 1).\beta \neq (HH')(\sigma 2).\beta$ [11]

14] $\sigma 1 \; {}^{\varphi}_{A}\Diamond^{HH'}_{\beta} \; \sigma 2$

Corollary 5-6

Given

1] φ is invariant

2] $\beta \notin A$

Prove: $A \; \mathbb{D}^{H}_{\varphi} \; \beta \supset$
$(\exists \delta, m \notin A)\,(\, A \; \mathbb{D}^{\delta}_{\varphi} \; m \,) \wedge (\exists \delta, M)\,(\, M \; \mathbb{D}^{\delta}_{\varphi} \; \beta \wedge \beta \notin M \,)$

by induction on the length of H

Base H = λ. Follows from [2, th 2-5]

Base H = δ. Direct by substitution

Induction Assume for H, prove for Hδ or δH

3] Assume $A \Vdash_{\varphi}^{H\delta} \beta$

4] $(\exists M)(A \Vdash_{\varphi}^{H} M \wedge M \Vdash_{\varphi}^{\delta} \beta)$ [3,1, th 5-4]

5] Case 1 $\beta \notin M$

6] $(\exists \delta, M)(M \Vdash_{\varphi}^{\delta} \beta \wedge \beta \notin M)$ [4,5]

7] Case 2 $\beta \in M$

8] $A \Vdash_{\varphi}^{H} \beta$ [4,7, th 5-3]

9] $(\exists \delta, M)(M \Vdash_{\varphi}^{\delta} \beta \wedge \beta \notin M)$ [1,2,8, induction]

10] Assume $A \Vdash_{\varphi}^{\delta H} \beta$

11] $(\exists M)(A \Vdash_{\varphi}^{\delta} M \wedge M \Vdash_{\varphi}^{H} \beta)$ [10,1, th 5-4]

12] Case 1 $\neg (M \subseteq A)$

13] $(\exists \delta, m \notin A)(A \Vdash_{\varphi}^{\delta} m)$ [11,12]

14] Case 2 $M \subseteq A$

15] $A \Vdash_{\varphi}^{H} \beta$ [13,14, th 2-2]

16] $(\exists \delta, m \notin A)(A \Vdash_{\varphi}^{\delta} m)$ [1,2,15, induction]

Q.E.D. [3-(5-6,7-9), 10-(12-13,14-16)]

Appendix B - References

[Ashby 56] W Ross Ashby, "An Introduction to Cybernetics", 1956

[Bell & LaPadula 73] D Bell, L J LaPadula, "Secure Computer Systems: A Mathematical Model", The Mitre Corp, MTR-2547, Nov 1973

[Case 74] K G Walter, etal, "Primitive Models for Computer Security", Dept Computer and Information Sciences, Case Western Reserve, ESD-TR-74-117

[Cohen 76] E Cohen, "Problems, Mechanisms & Solutions", PhD Thesis, CMU, Aug 1976

[Cohen & Jefferson 75] E Cohen, D Jefferson, "Protection in the HYDRA Operating System", Proceedings 5th Symposium on Operating System Principles, Nov 1975 (also SIGOPS, v9,5)

[Denning 75] D Denning, "Secure Information Flow in Computer Systems", PhD Thesis, Comp Sci Dept, Purdue Univ, May 1975

[Denning 76] D Denning, "A Lattice Model of Secure Information Flow", CACM v19,5 (May 1976)

[Floyd 67] R Floyd, "Assigning Meanings to Programs" in Proc of a Symp. in Applied Mathematics, Vol 19, Mathematical Aspects of Computer Science, J Schwartz (ed), AMS 1967

[Jones & Lipton 75] A Jones, R Lipton, "The Enforcement of Security Policies for Computations", 5th Symp. on Operating System Principles, Nov 1975

[Lampson 71] B Lampson, "Protection", 5th Annual Princeton Conference on Information Sciences and Systems, March 1971

[Lampson 73] B Lampson, "A Note on the Confinement Problem", CACM v16,10 (Oct 1973)

[Lipton 73] R Lipton, "On Synchronization Primitive Systems", Yale CSRR 22, 1973 (also CMU PhD Thesis)

[Millen 76] J K Millen, "Security Kernal Validation in Practice", CACM v19,5 (May 1976)

[Rotenberg 73] L Rotenberg, "Making Computers Keep Secrets", PhD Thesis MIT, MAC-TR-116, Sept 1973

[Shannon & Weaver 49] C Shannon, W Weaver, "The Mathematical Theory of Communication", U Illinois Press, 1949

[Weissman 69] Weissman C, "Security Controls in the Adept-50 Time-Sharing System" FJCC 1969

[Wulf 74] W Wulf, et.al., "HYDRA: The Kernel of a Multiprocess Operating System", CACM v17,6 (June 1974)