REPRINT

A Classical Automata Approach to Noninterference Type Problems

Ira S. Moskowitz and *Oliver L. Costich*

FROM:
Proceedings of the Computer Security Foundations Workshop 5, Franconia, NH, June 1992, pages 2-8, IEEE Press.

CONTACT:
Ira S. Moskowitz, Information Technology Division, Mail Code 5543, Naval Research Laboratory, Washington, DC 20375.

Oliver L. Costich, Information Technology Division, Contractor-Mail Code 5542, Naval Research Laboratory, Washington, DC 20375.

E-MAIL:
moskowit@itd.nrl.navy.mil
costich@itd.nrl.navy.mil

A Classical Automata Approach to Noninterference Type Problems

Ira S. Moskowitz

Center for Secure Information Technology
Code 5543
Naval Research Laboratory
Washington, DC 20375-5000

Oliver L. Costich*

Center for Secure Information Systems
George Mason University
Fairfax, VA 22030

Abstract

Using classical automata theory we show how noninterference can be viewed as a relatively simple phenomenon. We also give direction for future work concerning probabilistic security problems using classical automata theory.

1 Introduction

Many models have been proposed to model a secure computer system. Some of the representative early models are by Harrison *et al* [10], Denning [4], and the often mentioned Bell-LaPadula model [3]. Depending on how one interprets concepts such as "subject/user" and "object" it is not clear whether or not covert channels are taken into consideration in these models.

Noninterference [6, 7] was a concrete approach at preventing improper information flow in a deterministic system. Nondeducibility [21] was a more abstract attempt at looking at possible non-secure information flow in a secure system, i.e., a covert channel. Restrictiveness [12, 13] was ostensibly developed as a nondeterministic analog of noninterference to repair purported problems involved with "hooking up" secure systems. Probabilistic interference [8] arose to analyze situations in nondeterministic systems that could be interpreted probabilisticly. FM [15] and its successors [9] are other attempts to understand information flow via probability theory. Also, several authors (including these) have used probability theory to analyze covert channels via information theory.

Moskowitz previously investigated probabilistic channels using a technique similar to that for fibre bundles [17] to try to get a better theoretical handle on problems dealing with restrictiveness and probability. In this paper we propose a simpler model for understanding information flow. We use the techniques of classical automata theory, as first explicated by Rabin and Scott [19], to set up our model. To quote from the NCSC Integrity report [5, p. 75], "A significant advantage of having the model based on a standard notion like the automaton is that extensive literature and well-developed intuition become immediately applicable to the problem domain." Many of the non-standard state machine models used in previous models of computer security are rather complicated, and, we believe, unnecessarily so. Nature is not always beautiful, but most of the time it is. This is not to say that these models are incorrect or useless. On the contrary, they might be more useful, in certain cases, than what we propose when trying to apply the theory to actual situations. Our contention is that they are more than what is needed to understand the basic properties.

The descriptive power of the model that we describe in this paper is its ease of expression and its ability to capture deterministic, nondeterministic, and hopefully probabilistic situations in one simple model. Hopefully, this will be a useful tool for reasoning about security (to paraphrase McLean [14]). We feel that our deterministic model is the correct model for noninterference type properties. The nondeterministic model, that we construct in the manner of classical automata theory, is not the same as that for restrictiveness, as in [13]. An advantage of our models is that the nondeterministic model contains the deterministic one as a special case.

One beauty of classical automata theory is the way complex systems can be represented by a very simple model. One does not need to worry about outputs or internal events. If the automaton is carefully defined one need only concern oneself with inputs and the state changes that they induce. As shown in Arbib [1] we can always include outputs as part of the state of the system. Any internal events are in fact caused by earlier input taking the system to a certain state. We will expand on these ideas later in the paper. We note that Jacob, by using a category theoretic approach [11], has also expressed noninterference in a compact form.

2 Automata Theory

Our notation will roughly follow that of Bavel [2]. Given a finite set Σ, let Σ^* be the free monoid over Σ. In other words Σ^* is made up of all finite sequences (called strings or words) from elements of Σ.

*Supported by the Naval Research Laboratory under contract N0001489-C-2389.

The empty sequence is denoted by ϵ and acts as the multiplicative identity. The multiplication is given by concatenation.

Definition 1 *An automaton is a triple $A = (S, \Sigma, \delta)$, where*
(1) S is a finite set (of states)
(2) Σ is a nonempty finite set (the input alphabet)
(3) $\delta : S \times \Sigma^* \to S$ is a (transition) function satisfying $\forall s \in S$ and $\forall x, y \in \Sigma^*$, $\delta(s, xy) = \delta(\delta(s, x), y)$ and $\delta(s, \epsilon) = s$, where ϵ is the null sequence in Σ^*.

Notice that there is no mention of outputs in this definition. Some authors also include an initial state and a set of accepting (or final) states. Accepting states are not of interest to us here because we are not explicitly concerned with what languages the automaton recognizes. We are only concerned with observable state changes. Initial states will be discussed later.

We will first briefly discuss outputs and then see that they are not necessary.

Definition 2 *An automaton with outputs is the 5-tuple $M = (S, \Sigma, \delta, Y, \lambda)$, where S, Σ, and δ are as in Def. 1. The set Y is referred to as the set of outputs and λ is the output function $\lambda : S \times \Sigma \to Y$.*

Notice that the output is determined solely by the input and state. Also note that in this definition the output function does not change the state of the system. (This definition can be modified to look at output strings (or traces) by redefining λ so that $\forall s \in S$ and $\forall x, y \in \Sigma^*$, $\lambda(s, xy) = \lambda(s, x)\lambda(\delta(s, x), y)$.) It is the effect of inputs that concerns us. Modeling the outputs is superfluous as we shall see.

In [1] Arbib describes a state-output machine, which negates the necessity of our worrying about outputs as long as we choose our states properly.

Definition 3 *An automaton with outputs is a state-output machine if $\exists \beta : S \to Y$ such that $\lambda = \beta \circ \delta$.*

The function β need not always exist. It depends on how the system arrived at the state in question. In other words we may have an automaton with outputs such that $\lambda(s_1, x_1) = y_1$, $\lambda(s'_1, x'_1) = y'_1$, and $\delta(s_1, x_1) = \delta(s'_1, x'_1)$, but $y_1 \neq y'_1$. The function β does not exist in this case. The essence of β is that it attaches to the state s the output that goes along with it that resulted from an input taking a previous state into s.

It is an important fact that every automaton with output can be viewed as a state-output machine. To do this we replace every state s by the set $\hat{S} = \{[s, y] \mid \exists s' \in S, \exists x \in \Sigma$, such that $\delta(s', x) = s$ and $\lambda(s', x) = y\}$. Accordingly, we can define a new δ and λ, denoted by $\hat{\delta}$ and $\hat{\lambda}$, respectively by $\hat{\delta}([s, y], x) = [\delta(s, x), \lambda(s, x)]$ and $\hat{\lambda}([s, y], x) = \lambda(s, x)$. The function β in this case maps $[s, y] \to y$. Thus we may replace the entire set S by $\hat{S}$, where $\hat{S}$ is all such $[s, y]$ and use $\hat{\delta}$ and $\hat{\lambda}$. The important fact to keep in mind is that the input alphabet has not changed nor has the way inputs transition states really changed. This only fine-tunes the transitions to include the outputs along with the state changes. Due to this we view all automata with outputs as state-output machines (via the above construction). However, in a state-output machine the information about λ is superfluous because the outputs are incorporated into the states. That is, we need not consider the output function in reasoning about the security of such systems, since knowing the current state embodies knowing the current output.

The above justifies our use of automata as a model for our computer system. We do not have outputs or internal events causing state transitions, as do McCullough and others. This is because internal events and outputs are caused by inputs. In our thinking the system starts in a benign (not necessarily unique) initial state and inputs move it out of this initial state. Outputs arise from inputs and internal events are caused by previous input moving the system into a state that is prepared to transition to yet another state. One can consider the next state from an input which triggers internal events as the sequences of states passed through, which can be represented as a single state in our model. (We address models that view each internal transition as a separate state in the digression at the end of section 4.)

We do not model the time between state transitions. Our model only allows us to talk about before and after with respect to state transitions, and our previous comments show that this is sufficient for our present purposes. For security our concern is *can high inputs affect what low "sees"*. We are certainly not the first to be concerned with this issue. However, an aim of our model is to make the security issues more transparent and amenable to many kinds of analysis.

3 Carpe States

An assumption that must be made clear is exactly how Low "sees the states". A state is a vector of variable values. The values of the variables determine what state the system (automaton) is in. These variables, the objects of the secure system, are designated either low or high. The high user (High) knows the value of the entire state vector, whereas the low user (Low) knows only the values of the low variables. In other words, Low does not see complete states and therefore cannot distinguish between states which differ only in the values of the high variables. This corresponds to McCullough's treatment of states. Note that our states, by assumption of the state-output construction discussed in the previous section, include the outputs as state variables. Hence Low, in a sense, can see the low outputs while High can see all of the outputs. Of course implicit is that the users are able to determine the values of the appropriate variables by some assumed means.

Our systems are input total. This is reflected mathematically by having δ defined as a function, not a partial function. The transition function has as its domain $S \times \Sigma^*$. Any state and any input result in another (possibly the same) state. We view the automaton as starting in an initial state with all of the high variables "zeroed out". We would not want the automaton to start in an initial state whose high variables contained the instruction "`after the next three low inputs do something to the low state variables`". We need a secure beginning for the system. We need this to make sure that only inputs along with possibly prior low information already in the system can influence where the system is going. Our security concern becomes *high inputs cannot affect the low state variables.* We will make this precise later in the paper. Below is our definition for security, which is, of course, similar to Goguen and Meseguer's definition of noninterference [6, 7] and McCullough's state based definition of restrictiveness [12]. Elements of Σ in our model correspond to user-command pairs of Goguen and Meseguer, i.e., $\Sigma = U \times C$ where U = users and C = commands.

4 Deterministic Secure System Model

Definition 4 *We say that an automaton is* <u>*double level*</u> *if there are two and only two users designated as High and Low, and that the inputs come from either High or Low, but not both.*

We will assume for the rest of the paper that all of our automata are double level. We refer then to the inputs as low or high inputs. Notice then that Σ is the disjoint union of Σ_L (the low inputs) and Σ_H (the high inputs).

Definition 5 *We say that s_0 is a* <u>*secure initial state*</u> *if no high inputs have yet been entered into the machine.*

In other words, a state is a secure initial state if no high information has yet affected the high state variables.

Definition 6 *Given $s_1, s_2 \in S$ we say that s_1 and s_2 are equivalent, written $s_1 \sim s_2$, iff the low state variables of s_1 and s_2 have the same values.*

This is obviously an equivalence relation on S and we may form the quotient set $S/\!\sim$ and the quotient (or projection) map $\pi : S \rightarrow S/\!\sim$.

Definition 7 *Define $F : \Sigma^* \rightarrow \Sigma_L^*$, by $F(w) = F(x_1) \cdots F(x_n)$, where $w = x_1 \cdots x_n$ and each $x_i \in \Sigma \cup \{\epsilon\}$, and*

$$F(x_i) = \begin{cases} x_i & \text{if } x_i \in \Sigma_L \\ \epsilon & \text{if } x_i \in \Sigma_H \cup \{\epsilon\} \end{cases}.$$

Note that F is an onto homomorphism of free monoids and that F restricted to Σ_L^* is just the identity map. In fact F is just the purge map discussed by others [22]. All of this leads us to a definition of a Secure Deterministic Automaton (SDA).

Definition 8 (SDA) *Given a system represented by both an automaton $A = (S, \Sigma, \delta)$ and a secure initial state s_0, we say that the system is a Secure Deterministic Automaton if:*
(1) The map $\tilde{\delta} : S/\!\sim \times\, \Sigma_L^* \rightarrow S/\!\sim$, given by
$\tilde{\delta}([s], w_L) = [\delta(s, w_L)]$, where $[s] \in S/\!\sim$ and $w_L \in \Sigma_L^*$, is well-defined.
(2) The following diagram, referred to as the Deterministic Security Diagram (DSD) is commutative -

$$\begin{tikzcd} S \times \Sigma^* \arrow[r, "\delta"] \arrow[d, "\pi \times F"'] & S \arrow[d, "\pi"] \\ S/\!\sim \times \Sigma_L^* \arrow[r, "\tilde{\delta}"'] & S/\!\sim \end{tikzcd}$$

Let us analyze the DSD to see why we require such a definition for security. First of all, the map $\tilde{\delta}$ must be well-defined for a system to be secure. Suppose that $s_1 \sim s_2$, i.e., $[s_1] = [s_2]$. If $\delta(s_1, w_L) \not\sim \delta(s_2, w_L)$ then Low can tell, by inputting w_L, that s_1 and s_2 are different elements of S. This is a security violation for then Low would have more information than just the values of the low state variables! It is possible that High could manipulate the automaton into state s_1 or s_2 through a series of (not necessarily consecutive) inputs. This would allow a covert channel to be opened up between High and Low. Therefore $[\delta(s_1, w_L)] = [\delta(s_2, w_L)]$ is a requirement.

Secondly, the diagram must commute. Since F restricted to Σ_L^* is the identity map the only way that the diagram could not commute would involve an element of $x_H \in \Sigma_H$ and an $s \in S$ such that $\delta(s, x_H) \not\sim \delta(s, \epsilon) = s$. If High inputs can affect the equivalence class of a state then Low will know that High input something. Therefore this cannot be allowed.

The DSD also satisfies the Bell-LaPadula condition which forbids reads up and writes down. The no read up policy is enforced by the fact that Low has knowledge only of the Low state variables. Since we are not concerned with aggregation [16] problems all high information must come from high input. Since high input is not allowed to change low state variables and we started in a secure initial state no write down is also enforced.

Digression: Implicit in our construction of the automaton and description of the state variables is that the users do not see any of the intermediate processing of the state transitions. If intermediate states were in fact visible states to the users then internal transitions might be a problem in our model. Of course then δ would have to be modified to be a partial function and we would lose input totality. This is not a serious problem however, since a system with a partial function δ can be represented as a particular kind of nondeterministic automaton. We will see that these are behaviorally equivalent to (input total) deterministic ones. *End of Digression*

4.1 Unwinding

In the spirit of Goguen and Meseguer [7] we will examine an unwinding theorem.

Theorem 1 *A system is a SDA iff the DSD holds (is well-defined and commutes) with Σ^* and Σ_L^* replaced by Σ and $\Sigma_L \cup \epsilon$, respectively.*

Proof: Note that the map F never increases the length of strings.
(*Holds with* $* \Rightarrow$ *holds without* $*$) Trivial.
(*Holds with* $* \Leftarrow$ *holds without* $*$) This is a straightforward induction proof which we show only for commutativity. The fact that $\tilde{\delta}$ is well-defined follows in a similar (and simpler) fashion.
(a) - Commutativity holds for strings of length up to 1. This follows from the assumptions.
(b) - Assume commutativity holds for strings of length up to $n-1$. This is the induction hypothesis.
(c) - We show commutativity holds for strings of length n. Say $w \in \Sigma^*$ and $w = x_1 \cdots x_{n-1} \cdot x_n$, where each $x_i \in \Sigma$. Following the DSD diagram from the top to the right we have that $(s,\ x_1 \cdots x_{n-1} \cdot x_n) \xrightarrow{\delta} \delta(s,\ x_1 \cdots x_{n-1} \cdot x_n) \xrightarrow{\pi} [\delta(s,\ x_1 \cdots x_{n-1} \cdot x_n)]$. We have to follow the diagram from the left to the bottom and see if we get the same thing. Hence, $(s,\ x_1 \cdots x_{n-1} \cdot x_n) \xrightarrow{\pi \times F} ([s],\ F(x_1) \cdots F(x_{n-1}) \cdot F(x_n)\ ) \xrightarrow{\tilde{\delta}} [\delta(s,\ F(x_1) \cdots F(x_{n-1}) \cdot F(x_n)]$. So we now need to show that

$$\delta(s,\ F(x_1) \cdots F(x_{n-1}) \cdot F(x_n)\ ) \sim \delta(s,\ x_1 \cdots x_{n-1} \cdot x_n). \tag{1}$$

By the definition of δ we see that $\delta(s,\ x_1 \cdots x_{n-1} \cdot x_n) = \delta(\delta(s,\ x_1 \cdots x_{n-1}),\ x_n)$ and $\delta(s,\ F(x_1) \cdots F(x_{n-1}) \cdot F(x_n)\ ) = \delta(\delta(s,\ F(x_1) \cdots F(x_{n-1})\ ),\ F(x_n)\ )$. By step (b) of the induction we are assuming that $\delta(s,\ F(x_1) \cdots F(x_{n-1})\ ) \sim \delta(s,\ x_1 \cdots x_{n-1})$. Therefore we may apply step (a) of the induction to get equation (1) above. ∎

5 Nondeterminism

McCullough looked at nondeterministic systems when he defined restrictiveness, though his model is computationally different that ours. Our model easily generalizes to the nondeterministic case in the spirit of classical automata theory. We use the notation $\wp(X)$ for the power set of X.

Definition 9 *A <u>nondeterministic automaton</u> is a triple $A = (S, \Sigma, \delta)$, where*
(1) S is a finite set (of states)
(2) Σ is a finite nonempty set (the input alphabet)
(3) $\delta : S \times \Sigma^* \to \wp(S)$ is a (transition) relation satisfying $\forall s \in S$ and $\forall x, y \in \Sigma^*$, $\delta(s, xy) = \cup\{\delta(s', y) \mid s' \in \delta(s, x)\}$ and $\delta(s, \epsilon) = s$, where ϵ is the null sequence in Σ^*.

In other words given a state s and a word w, $\delta(s, w)$ is a subset of S. The transition relation $\delta(s, w)$ can be thought of as a set of triples $\{(s, w, s_1), (s, w, s_2), \cdots, (s, w, s_n)\}$. Each s_i is a state to which the automaton might transition, given that the automaton is presently in state s and w is the input string. We can, as before, obtain a security diagram by replacing S by $\wp(S)$ and generalizing the equivalence relation $\sim$.

Definition 10 *If $A, B \in \wp(S)$ we say that A is equivalent to B, written $A \sim B$, iff for each $s_A^i \in A$ $\exists s_B^j \in B$ such that $s_A^i \sim s_B^j$ and visa versa. Where $\sim$, with respect to states, is as before in the deterministic case.*

All we are doing is extending the definition of $\sim$ so that instead of state equivalence we can also talk about equivalent subsets of S. We use $\sim$ for both relations, letting context be the arbiter. Since $\sim$ is an equivalence relation on $\wp(S)$, we can define the quotient set $\wp(S)/\sim$ and the quotient mapping $\wp(\pi) : \wp(S) \to \wp(S)/\sim$.

Theorem 2 *There is a 1-1 and onto mapping between $\wp(S)/\sim$ and $\wp(S/\sim)$.*

Proof: The map π induces a map $\bar{\pi} : \wp(S) \to \wp(S/\sim)$ as follows. If $A \in \wp(S)$, $A = \{a_1, \cdots, a_n\}$ then $\bar{\pi}(A) = \{[a_{i_1}], \cdots, [a_{i_m}]\}$, where $a_{i_j} \in A$ and each $a_k \in A$ is equivalent to one and only one of the a_{i_j}. This map is clearly onto.

Suppose $A \sim B$, (in other words $[A] = [B]$ as elements of $\wp(S)/\sim$), where $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_m\}$. Consider $[s] \in \bar{\pi}(A)$. Then there is an $a_k \sim s$, $a_k \in A$ such that $\exists b_j \in B$ such that $[a_k] = [b_j]$. Therefore $[s] = [a_k] = [b_j] \in \bar{\pi}(B)$. Similarly we can show that $\bar{\pi}(B) \subset \bar{\pi}(A)$. We conclude that $\bar{\pi}(A) = \bar{\pi}(B)$. Hence we can "push" the map $\bar{\pi}$ down to $\wp(S)/\sim$ and induce a map ϕ which is well-defined.

$$\begin{tikzcd}
\wp(S) \arrow[r, "\bar{\pi}"] \arrow[d, "\wp(\pi)"'] & \wp(S/\sim) \\
\wp(S)/\sim \arrow[ur, "\phi"'] &
\end{tikzcd}$$

Suppose that $\bar{\pi}(A) = \bar{\pi}(B)$. Then given $s \in A$ $\exists s' \in B$ for which $s' \sim s$ and visa versa. Hence $[A] = [B]$, therefore ϕ is 1-1. Since $\bar{\pi}$ is onto so is ϕ. ■

Because of this mapping we can freely interchange between $\wp(S)/\sim$ and $\wp(S/\sim)$ througout the rest of the paper.

Example
$A = \{s_A^1, s_A^2, s_A^3, s_A^4\}$, $B = \{s_B^1, s_B^2, s_B^3\}$, $s_A^1 \sim s_A^4 \sim s_B^2 \sim s_B^3$, and $s_A^2 \sim s_A^3 \sim s_B^1$. So $A \sim B$ and $[A] = \{[s_A^1], [s_A^2]\}$.

As in the deterministic case we have the well known paradigm that a high input should not change the state to a nonequivalent state. Also Low should not be able to distinguish between subsets of $\wp(S)/\sim$. In the deterministic case our concern was with equivalence classes of elements. Now in the nondeterministic case we have to be concerned with sets of equivalence classes.

Example
Consider the states $a, b, b', c, c', d, d', e, e' \in S$ such that $b \sim b'$, $c \sim c'$, $d \sim d'$, but $e \not\sim e'$. Consider $w \in \Sigma_L^*$, $w = x_L^1 x_L^2$. Suppose x_L^1 can take a to either b, b', c or c'; and x_L^2 can take b to d and c to e, and x_L^2 can take b' to d' and c' to e'. On the quotient level Low sees two different possible transition scenarios

$\{[a]\} \xrightarrow{x_L^1} \{[b], [c]\} \xrightarrow{x_L^2} \{[d], [e]\}$ or

$\{[a]\} \xrightarrow{x_L^1} \{[b'], [c']\} \xrightarrow{x_L^2} \{[d], [e']\}$.

The above example is certainly not good from a security point of view. Low can know that there is a difference between $\{[b], [c]\}$ and $\{[b'], [c']\}$ because the same input string takes one to non-equivalent states. This in turn means that the sequence of states passed through must be different. But they were supposed to be equivalent to the low user.

From our above discussion and example we see that the analog of SDA to the nondeterministic case is the following:

Definition 11 (SNA) *Given a nondeterministic system represented by both a nondeterministic automaton $A = (S, \Sigma, \delta)$ and a set of secure initial states S_0, we say that the system is a Secure Nondeterministic Automaton (SNA) if:*
(1) The map $\tilde{\delta} : S/\sim \times \Sigma_L^* \rightarrow \wp(S)/\sim$, given by $\tilde{\delta}([s], w_L) = [\{\delta(s, w_L)\}]$, where $[s] \in S/\sim$ and $w_L \in \Sigma_L^*$, is well-defined.
(2) The following diagram, referred to as the Nondeterministic Security Diagram (NSD) is commutative -

$$\begin{tikzcd}
S \times \Sigma^* \arrow[r, "\delta"] \arrow[d, "\pi \times F"'] & \wp(S) \arrow[d, "\wp(\pi)"] \\
S/\sim \times \Sigma_L^* \arrow[r, "\tilde{\delta}"'] & \wp(S)/\sim
\end{tikzcd}$$

Of course we get a similar version of the unwinding theorem for the nondeterministic case.

Theorem 3 *A system is a SNA iff the NSD holds (is well-defined and commutes) with Σ^* and Σ_L^* replaced by Σ and $\Sigma_L \cup \epsilon$, respectively.*

Proof: Follows just as in the deterministic case. ■

Of course every deterministic system is a special case of a nondeterministic system, i.e., the transitions map into the singleton subsets of $\wp(S)$. Hence, the above definition of NSA actually includes DSA as a special case. Hence, we can see then how noninterference generalizes in the nondeterministic case.

Finally, we note that the subset construction of Rabin and Scott [19] can be applied to the nondeterministic systems presented here. Given a SNA, $A = (S, \Sigma, \delta)$ with a set of initial states S_0, we can define a SDA, $\wp(A) = (\wp(S), \Sigma, \wp(\delta))$ with initial state S_0 by defining $\wp(\delta) : \wp(S) \times \Sigma^* \rightarrow \wp(S)$ by $\wp(\delta)(B, x) = \cup\{\delta(b, x) \mid b \in B\}$ for $x \in \Sigma$ and using a recursive definition to extend $\wp(\delta)$ to Σ^*. It is well known that the resulting SDA $\wp(A)$ completely mimics the behavior of A with respect to state transitions and sets of input strings. For systems with output or accepting states this behavior is also captured. We state without proof:

Theorem 4 *Every SNA is behaviorally equivalent to an SDA with the same input set Σ^*.*

<u>Corollary</u>: The requirement of input totality for SDA's can be removed with no loss of generality in the model.

Proof: Let $A = (S, \Sigma, \delta)$ be an SDA except that $\delta : S \times \Sigma \to S$ is a partial function. Construct $\wp(A)$ and notice that $\wp(\delta)$, when restricted to $\{(\{s\}, x) \mid s \in S, x \in \Sigma\}$ yields either a singleton or the empty set. Thus $\wp(A)$ has the same behavior as the given SDA with partial transition function δ. Moreover, $\wp(A)$ has the same behavior as a deterministic, input total machine. ∎

6 Future Work

We believe that the commutative diagram approach that we started in [17] and continued here can give a simple model for detecting probabilistic channels. In fact, the NSD diagram should have the obvious analog in the probabilistic case. By this we mean a covert channel that arises by High and Low knowing the probabilities associated with nondeterministic transitions. Of course this all boils down to Shannon's [20, 18] analysis of discrete noisy channels. We feel, similar to McLean [15], that a simple conditional probability statement should suffice to show that the bandwidth of the probabilistic channels is zero. Of course we feel that only input strings need to be considered in the conditional probabilities. Gray [9] has looked at similar ideas but his model involved more complex alphabets than just input strings; however, as we have discussed we feel that modeling in this detail is too complex. For an abstract tool, we feel that an Ockham's razor approach is the most fruitful.

We also plan to discuss various compositions of SDA's and timing channels in future research.

Acknowledgements

We wish to thank Myong Kang, John McDermott, John McLean, and an anonymous referee for their helpful comments.

References

[1] Michael A. Arbib. *Theories of Abstract Automata.* Prentice-Hall, Englewood Cliffs, NJ, 1969.

[2] Zamir Bavel. *Introduction to The Theory of Automata.* Reston, Reston, VA, 1983.

[3] D.E. Bell and L.J. La Padula. *Secure Computer System: Unified Exposition and Multics Interpretation, MTR-2997.* MITRE Corp., Bedford, MA, March 1976.

[4] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[5] Department of Defense, National Computer Security Center. *Integrity in Automated Information Systems, C Technical Report 79-91*, September 1991.

[6] Joseph A. Goguen and José Meseguer. Security policies and security models. In *Proc. of the 1982 IEEE Computer Society Symposium on Computer Security and Privacy*, pages 11–22, Oakland, CA, 1982.

[7] Joseph A. Goguen and José Meseguer. Unwinding and inference control. In *Proc. of the 1984 IEEE Computer Society Symposium on Computer Security and Privacy*, pages 75–86, Oakland, CA, 1984.

[8] James W. Gray, III. Probabilistic interference. In *Proc. of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 170–179, Oakland, CA, May 1990.

[9] James W. Gray, III. On information flow security models. In *Proc. of the 1991 Workshop on Computer Security Foundations*, pages 55–60, Franconia, NH, 1991.

[10] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[11] Jeremy Jacob. Categorising non-interference. In *Proc. of the 1990 Workshop on Computer Security Foundations*, pages 44–50, Franconia, NH, 1990.

[12] Daryl McCullough. *FOUNDATION OF ULYSSES: The Theory of Security, Interim Report RADC-TR-87-222.* Odyssey Research Associates, Inc., Ithaca, NY, July 1988.

[13] Daryl McCullough. A hookup theorem for multilevel security. *IEEE Transactions on Software Engineering*, 16(6):563–568, June 1990.

[14] John McLean. Reasoning about security models. In *Proc. of the 1987 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 123 – 131, Oakland, CA, April 1987.

[15] John McLean. Security models and information flow. In *Proc. of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 180–187, Oakland, CA, May 1990.

[16] Catherine Meadows. Extending the Brewer-Nash model to a multilevel context. *Proc. of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 95–102, May 1990.

[17] Ira S. Moskowitz. Quotient states and probabilistic channels. In *Proc. of The Computer Security Foundations Workshop III*, pages 74–83, Franconia, NH, June 1990.

[18] Ira S. Moskowitz and Allen R. Miller. The channel capacity of a certain noisy timing channel. *IEEE Transactions on Information Theory*, 38(4):1339–1344, July 1992.

[19] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal*, pages 114–125, April 1959.

[20] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, IL, 1949. Also appeared as a series of papers by Shannon in the Bell System Technical Journal, July 1948, October 1948 (A Mathematical Theory of Communication), January 1949 (Communication in the Presence of Noise).

[21] David Sutherland. A model of information. In *Proc. of the 9th National Computer Security Conference*, pages 175–183, September 1986.

[22] J. Todd Wittbold. Controlled signalling systems and covert channels. In *Proc. of The Computer Security Foundations Workshop II*, pages 87–104, Franconia, NH, June 1989.