A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions*

John McLean
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington D.C. 20375

Abstract

This paper presents a general theory of system composition for "possibilistic" security properties. We see that these properties fall outside of the Alpern-Schneider safety/liveness domain and hence, are not subject to the Abadi-Lamport Composition Principle. We then introduce a set of trace constructors called *selective interleaving functions* and show that possibilistic security properties are closure properties with respect to different classes of selective interleaving functions. This provides a uniform framework for analyzing these properties and allows us to construct a partial ordering for them. We present a number of composition constructs, show the extent to which each preserves closure with respect to different classes of selective interleaving functions, and show that they are sufficient for forming the general hook-up construction. We see that although closure under a class of selective interleaving functions is generally preserved by product and cascading, it is not generally preserved by feedback, internal system composition constructs, or refinement. We examine the reason for this.

1 Introduction

The ability to build systems that satisfy a given property from a selected set of specified components is a requisite for the production of networks, the production of systems using off-the-shelf products, and the production of systems from verified components. However, a general ability to build composite high-assurance systems presupposes a general theory of system composition. Such a theory provides insight into why certain properties are preserved or not preserved by certain forms of composition. More importantly, for a large class of properties and a variety of composition constructs, it answers questions of the form: "If a system satisfying property X is composed with a system satisfying property Y using composition construct Z, what properties will the composite system satisfy?".

A general theory of system composition is clearly lacking for confidentiality properties. We know that Restrictiveness [8] and Noninference [14, 16] are preserved by general composition or *hookup*[1], that Nondeducibility on Strategies [17] is preserved by asynchronous composition [15], and that many properties are not preserved by general composition. However, we know nothing about the composability of Restrictiveness, Noninference, or Nondeducibility on Strategies with properties besides themselves, and we know nothing about the composability of other properties beyond the fact that they are not preserved by general composition with themselves. For example, we do not know what properties would be satisfied by a system in which a component satisfying Deducibility Security [19] was cascaded with a component satisfying Restrictiveness. As a result, we use Restrictiveness or Noninference in cases where better properties (either simpler and just as secure in the case of Restrictiveness, or just as simple yet more secure in the case of Noninference) may work. As new properties are developed, the situation will deteriorate further.

For this reason general theories of system composition, such as the one developed by Abadi and Lamport [1], are extremely appealing. A number of researchers in the security community are attempting to use the Abadi-Lamport Composition Principle to develop a general theory of composition for confidentiality properties. However, the Abadi-Lamport Composition Principle is restricted to the class of properties that are definable within the safety/liveness property framework originally presented by Alpern and Schneider in [2]. Since "possibilistic" security properties (a class of properties which includes Generalized Noninterference, Restrictiveness, Noninference, Nondeducibility on Strategies, and Deducibility Security) fall outside this domain, the Abadi-Lamport Composition Principle is not directly applicable.

This paper presents a general theory of system composition for a class of "possibilistic" properties. In *Section 2* we introduce a system model and a set of trace constructors called *selective interleaving functions*. The model space, an instantiation of the Alpern and Schneider framework, is extendible to probabilis-

*Forthcoming in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy.*

[1]Given two systems, their *hookup* is the composite system where each component system can communicate (receive input from and send output to) with both the other component system and the outside world.

tic model spaces, e.g., as found in [5]. We consider the standard possibilistic security properties and two new ones: Generalized Noninference, which is an extension of Noninference, and Separability, which has affinities both to Rushby's Separation Kernel [18] and to Nondeducibility on Strategies. We show that all of these properties are closure properties with respect to classes of selective interleaving functions. This provides a uniform framework for analyzing such properties and allows us to construct a partial ordering for them.

In *Section 3.1* we present three external composition constructs: product, cascade, and feedback. We show the extent to which each of these preserves closure with respect to different classes of selective interleaving functions and show that product and feedback are sufficient for forming the general hook-up construction. We see that Separability provides a composable alternative to Restrictiveness and Noninference, which is simpler than the former and more secure than the latter.

In particular we shall see that the product of two systems behaves quite well with respect to security properties. Further, when two systems are cascaded:

- Separability is preserved when composed with itself;
- Noninference is preserved when composed with itself and with Separability;
- Generalized Noninterference is preserved when composed with itself and with Separability; and
- Generalized Noninference is preserved when composed with itself, with Noninference, with Generalized Noninterference, and with Separability.

We shall also see that when two systems are composed with a feedback construction:

- Separability is preserved when composed with itself; and
- Noninference is preserved when composed with itself and with Separability.

The extent to which other properties are preserved when composed with the feedback construction depends on the particulars of the system. In *Section 3.2* we see that this is also true for internal composition (union, intersection, and set difference) and for refinement. In *Section 4* we shall shall gain some insight into why feedback and internal composition causes problems for possibilistic security properties.

This paper is not meant to be an argument for using possibilistic security models. I have discussed the limitations of such models elsewhere [10, 13] and shall not re-visit these issues here. However, when compared to their probabilistic counterparts, such as [5, 10], possibilistic security models provide us with a relatively simple model for building systems and have, for this reason, enjoyed a great deal of popularity. This paper is an attempt to understand these models and their composition more thoroughly and to provide better alternatives to the models than are currently available.

2 System Model and System Properties

In *Section 2.1* we define the notion of a system state and use this definition to present the Alpern-Schneider concepts of a property, of a system, and of a property holding for a system. We also see how these concepts are embedded in the Abadi-Lamport concepts of a specification and of a system satisfying a specification. We then examine the limitations of the Alpern-Schneider framework for analyzing possibilistic security properties and their composition. In *Section 2.2* we extend the Alpern-Schneider concept of a property by introducing the trace set property of being closed under a class of selective interleaving functions. This provides a framework for examining possibilistic security properties. We go on to establish some elementary facts about such properties and their relationships.

2.1 The Alpern-Schneider Framework and Its Limitations

The Alpern-Schneider framework is transparent with respect to any particular notion of system state. To make things more concrete, we introduce the following characterization:

Definition 2.1 (State Space) For nonnegative integers m and n, let $\langle in_1, ..., in_m\rangle$ be a tuple of m distinct *input variables* and $\langle out_1, ..., out_n\rangle$ be a tuple of n distinct *output variables* such that the *ith* input variable ranges over some alphabet I_i and the *ith* output variable ranges over some alphabet O_i. A *state space* is the set $\{\langle\langle in_1, ..., in_m\rangle, \langle out_1, ..., out_n\rangle\rangle |\ in_i \in I_i \wedge out_i \in O_i\}$. An element of a *state space* is called a *system state.* □

As an example, consider the state space whose states are of the form $\langle\langle in_1, ..., in_m\rangle, \langle out_1, ..., out_m\rangle\rangle$ where for all $1 \leq i \leq m$: $I_i = O_i = \{0, 1, \lambda\}$. Assume that for some $1 \leq n < m$: $in_1, ..., in_n$ and $in_{n+1}, ..., in_m$ are input channels that contain the inputs of $H = n$ distinct high-level users and $L = m - n$ distinct low level users, respectively, and that $out_1, ..., out_n$ and $out_{n+1}, ..., out_m$ are output channels that contain the outputs to these same users. Of course, some of the high-level users may be Trojan Horses operating on behalf of some of the low-level users. If there is no current input or output on a particular channel, the channel takes on the value λ. In the future we shall refer to this state space with H high-level users and L low-level users as the *two level security state space*, which we denote by $\hat{\mathbf{\Sigma}}$. We shall denote $\langle in_1, ..., in_n\rangle$, $\langle in_{n+1}, ..., in_m\rangle$, $\langle out_1, ..., out_n\rangle$, and $\langle out_{n+1}, ..., out_m\rangle$ by *highin*, *lowin*, *highout*, and *lowout*, respectively.

Notation for Tuples: Given a set σ, we shall use the notation σ^n to denote σ's nth-iterated Cartesian product $\sigma \times ... \times \sigma$, and given the symbol α, we shall use α^n to denote the n-tuple $\langle \alpha, ..., \alpha \rangle$. Given tuples $x = \langle x_1, ..., x_m \rangle$ and $y = \langle y_1, ..., y_n \rangle$, we shall use $x[i]$ to denote x_i and $\langle x : y \rangle$ to denote $\langle x_1, ..., x_m, y_1, ..., y_n \rangle$. □

Definition 2.2 (Trace Set) Given a state space, Σ, Σ's *trace space*, written $trace(\Sigma)$, is the set $\{\langle s_1, s_2, ... \rangle | s_i \in \Sigma\}$. An element of a trace space is called a *trace*. A subset of a trace space is called a *trace set*. A *trace set* $\sigma 1$ is a *refinement* of a *trace set* $\sigma 2$ if and only if $\sigma 1 \subseteq \sigma 2$. □

As an example, consider $\hat{\mathbf{\Sigma}}$ introduced above. The trace space $trace(\hat{\mathbf{\Sigma}})$ is the set of traces of the form

$$\begin{aligned} t &= \langle \langle \langle highin_1 : lowin_1 \rangle, \langle highout_1 : lowout_1 \rangle \rangle, \\ &\quad \langle \langle highin_2 : lowin_2 \rangle, \langle highout_2 : lowout_2 \rangle \rangle, ... \rangle, \end{aligned}$$

where $highin_i$, $lowin_i$, $highout_i$, and $lowout_i$ represent the high-level and low-level input to and output from the system at time i.

By eliminating traces, refinements of $trace(\hat{\mathbf{\Sigma}})$ can reduce nondeterminism and limit the input domain. However, since refinements cannot introduce new behaviors, any property that is satisfied by every trace of a subset of $trace(\hat{\mathbf{\Sigma}})$ is preserved by every trace of any refinement of that subset. So, for example, if every trace in some trace set $\sigma \subseteq trace(\hat{\mathbf{\Sigma}})$ has the proprety that $highout_i = highin_i + lowin_i$, then every trace in any refinement of σ also has this property.

As another example of a trace set we shall find useful, consider the state space $\{\langle \langle in \rangle, \langle out \rangle \rangle | \ in \in I \wedge out \in O\}$ where $I = O$. We shall call the trace set $\hat{\mathbf{I}} = \{\langle \langle \langle in_1 \rangle, \langle out_1 \rangle \rangle, \langle \langle in_2 \rangle, \langle out_2 \rangle \rangle, ... \rangle | \ in_i = out_i\}$ the *identity system*.

Notation for Traces: Given a trace

$$\begin{aligned} t &= \langle \langle \langle in^1_1, ..., in^1_j \rangle, \langle out^1_1, ..., out^1_k \rangle \rangle, \\ &\quad \langle \langle in^2_1, ..., in^2_j \rangle, \langle out^2_1, ..., out^2_k \rangle \rangle, ... \rangle, \end{aligned}$$

we shall use the following notational conventions:

$$\begin{aligned} t[i] &= \langle \langle in^i_1, ..., in^i_j \rangle, \langle out^i_1, ..., out^i_k \rangle \rangle; \\ t[i...n] &= \langle \langle \langle in^i_1, ..., in^i_j \rangle, \langle out^i_1, ..., out^i_k \rangle \rangle, ..., \\ &\quad \langle \langle in^n_1, ..., in^n_j \rangle, \langle out^n_1, ..., out^n_k \rangle \rangle \rangle; \\ in(t) &= \langle \langle in^1_1, ..., in^1_j \rangle, \langle in^2_1, ..., in^2_j \rangle, ... \rangle; \\ out(t) &= \langle \langle out^1_1, ..., out^1_k \rangle, \langle out^2_1, ..., out^2_k \rangle, ... \rangle; \\ in(t)[l...m] &= \langle \langle in^l_1, ..., in^l_j \rangle, ..., \langle in^m_1, ..., in^m_j \rangle \rangle; \\ out(t)[l...m] &= \langle \langle out^l_1, ..., out^l_k \rangle, ..., \langle out^m_1, ..., out^m_k \rangle \rangle; \\ in[l...m](t) &= \langle \langle in^1_l, ..., in^1_m \rangle, \langle in^2_l, ..., in^2_m \rangle, ... \rangle; \\ out[l...m](t) &= \langle \langle out^1_l, ..., out^1_m \rangle, \langle out^2_l, ..., out^2_m \rangle, ... \rangle; \end{aligned}$$

In the case of $trace(\hat{\mathbf{\Sigma}})$, we shall use $highin(t)$, $lowin(t)$, $highout(t)$, and $lowout(t)$ to refer to $in[1...n](t)$, $in[(m-n)...m](t)$, $out[1...n](t)$, and $out[(m-n)...m](t)$, respectively. □

Following Alpern and Schneider, a *property* and a *system* are both trace sets, and a property *holds for* a system if and only if the system is a refinement of the property [2]. Intuitively, a property trace set consists of those traces that satisfy the property and a system trace set consist of those traces that the system can exhibit. Abadi and Lamport add to this framework the concept of a *specification*, which is a property formed by taking the union of the set of traces that conform to a system's desired behavior and the set of traces that contain violations of a system's input restrictions [1]. The latter set reflects assumptions about the environment in which the system is to be run. The former set reflects requirements about how a system can react when placed in an environment that satisfies its input restrictions. A program *satisfies* a specification if the specification holds for the program.

The Alpern-Schneider framework is very appealing. The conception of property as a set of traces has the theoretical consequence of making every property the intersection of a safety property and a liveness property [2], and the conception of an implementation as refinement seems very natural given the fact, noted above, that refinement preserves properties of traces. Further, the ability to specify input restrictions makes it unnecessary to reason about a system's reaction to an environment that fails to satisfy its restrictions. This is in contrast to the assumption of input totality usually made in the security community, for example, in [8, 19]. Finally, the Abadi-Lamport Composition Principle makes it possible to determine from component specifications whether or not a composite comprising those components satisfies its specification.

A limitation of the Alpern-Schneider framework is that not every system property of interest is a property of traces. For example, Abadi and Lamport note that average response time over all possible executions is not a property of traces. They do not seem to regard this as a serious limitation of the Alpern-Schneider framework, however, since there is a trace property that approximates it (*viz.*, average response time over long sequences of events within a single trace) [1].

However, there are system properties for which it is unclear that such "nice" trace-level approximations exist. For example, consider a multi-level system that takes a set of integers as input in_i and returns some permutation of the set as output out_i. Confidentiality considerations may lead to the requirement that the permutation a low-level user sees cannot be affected by high-level input (i.e., any legal low-level permutation is co-possible with any legal high-level input). Integrity considerations may lead to the requirement that the permutation that a high-level user sees cannot be affected by low-level input (i.e., any legal high-level permutation is co-possible with any legal low-level input). Availability considerations may lead to the requirement that if a system's high-level response time slows down, the delay cannot have been caused by low-level behavior (i.e, any legal high-level delay must be co-possible with any legal low-level input).

The fact that possibilistic properties are not properties of traces follows immediately from the fact that they are not preserved by trace subsetting [12].[2] For example, consider the two user security state space, $\hat{\mathbf{\Sigma}}$, and the confidentiality property P that any legal low-level behavior must be co-possible with all legal high-level behaviors. If P were a property of traces, there would be a set σ consisting of those traces of $trace(\hat{\mathbf{\Sigma}})$ that satisfy P and systems would satisfy P only in the sense that they were subsets of σ. Since a system $\sigma 1$ consisting of all traces trivially satisfies P, $\sigma 1$ would be a subset of σ. However, a system $\sigma 2$ consisting of those traces t of $\sigma 1$ in which high-level input $highin(t)[i]$ is echoed as low-level output $lowout(t)[i+1]$ does not satisfy P and, therefore, would not be a subset of σ. Hence, if P were a property of traces, we would be faced with the contradiction that $\sigma 1$ would be a subset of σ, yet $\sigma 2$ would not be a subset of σ even though $\sigma 2 \subseteq \sigma 1$. Similar arguments apply for each of the properties listed above since each requires that a system must exhibit certain behaviors (not in the liveness sense of saying that the behavior must eventually happen, but in the possibilistic sense that the system could have done otherwise).

Nor do these properties seem to have "nice" trace-level properties that approximate them. For example, it may be possible to form a trace-level approximation by borrowing techniques from the theory of Kolmogorov complexity and say that a trace is secure if knowledge of its low-level events does not help us to determine its high-level input [7]. However, such an approach would clearly sacrifice the relative simplicity possibilistic security models enjoy over their probabilistic counterparts [5, 10].

Although the fact that these security properties are not preserved by refinement implies the fact that these properties are not properties of traces, the two points are distinct and deserve to be separated. Returning to property P, the former point shows that functionally correct implementations of specifications that satisfy P do not necessarily preserve P.[3] The latter point is more fundamental. It shows that P is not definable within the Alpern-Schneider framework to begin with. Hence, we may be able to write specifications that satisfy P, but we cannot reason about them or their composition within the Alpern-Schneider framework. Nor can we apply composition principles, such as Abadi and Lamport's [1], that are limited to Alpern-Schneider properties.

[2] The fact that many confidentiality properties are not preserved by the standard notion of refinement has been noted by McCullough [8] and addressed, to some extent, in [4], [6], [11], and *Section 3.2* of this paper. The fact that these properties are not trace sets is a distinct point, first pointed out to me by Jim Gray, although Gray's original argument differs from the one presented here.

[3] This is not simply because lower level implementation detail may introduce new channels, but because elimination of possible system output may turn zero capacity channels into positive capacity channels.

2.2 Security Models and Selective Interleaving Functions

If possibilistic security properties are not properties of traces, i.e., trace sets, what are they? The answer is that they are properties of trace sets, i.e., sets of trace sets. For example, consider the purge function that sets all high-level input and output in a trace t to λ, i.e., the function $purge : trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$, such that

$$purge(t) = \langle\langle\langle \lambda^H : lowin(t)[1]\rangle, \langle \lambda^H : lowout(t)[1]\rangle\rangle, \langle\langle \lambda^H : lowin(t)[2]\rangle, \langle \lambda^H : lowout(t)[2]\rangle\rangle, \ldots\rangle.$$

Noninference, originally due to O'Halloran [16], is the property that is satisfied by a trace set σ if and only if σ is closed under $purge$.[4]

For deterministic systems, Noninference is equivalent to Goguen and Meseguer's Noninterference [3] if we assume that high-level output cannot be generated when there is no high-level input. Hence, for deterministic systems satisfying this assumption, Noninference shares Noninterference's property of being practically perfect [10]. Further, Noninference is more general than Noninterference in that the latter fails to be directly applicable to nondeterministic systems. However, as noted in [13], Noninference is too strong for systems in which high-level output can exist without high-level input and too weak, in general, since it allows low-level output to be influenced by the *insertion* of high-level input.

By generalizing the notion of *purge*, we obtain a nondeterministic formulation of Noninterference that does not contain the assumption that high-level output can be generated only when there is high-level input. Say that $f : trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$ is an *input purge* if and only if $f(s) = t$ implies that $highin(t) = \langle \lambda^H, \lambda^H, \lambda^H, \ldots\rangle$, $lowin(t) = lowin(s)$, and $lowout(t) = lowout(s)$. In other words, a function f is an input purge if it sets all high-level inputs to λ and does not alter low-level inputs or outputs. Two input purges may differ in what they assign to high-level outputs, however. For example, the function *purge* defined above is the input purge that sets all high-level outputs to λ, but there are other input purges. Say that a system satisfies *Generalized Noninference* if and only if the system is closed under some input purge.

A formulation of Noninterference that does not employ purge functions but, instead, a more general concept of trace interleaving is derivative of Sutherland's notion of Deducibility Security [19]. Consider the function $interleave : trace(\hat{\mathbf{\Sigma}}) \times$

[4] A set σ is closed under a function f if and only if $s \in \sigma$ implies that $f(s) \in \sigma$. By analogy, we shall extend the notion to multi-argument functions. For example, σ is closed under $f : \sigma \times \sigma \rightarrow \sigma$ if and only if $s1 \in \sigma$ and $s2 \in \sigma$ implies that $f(s1, s2) \in \sigma$.

$trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$ such that $interleave(t1, t2) = t$ implies that $highin(t) = highin(t1)$, $lowin(t) = lowin(t2)$, $highout(t) = highout(t1)$, and $lowout(t) = lowout(t2)$. Say that a system satisfies *Separability* if and only if it is closed under *interleave*. Separability is preferable to Sutherland's Deducibility Security, which requires only that a high-level history can be inserted somewhere in a low-level history, since Deducibility Security is extremely weak [13]. In fact, in many ways Separability more closely resembles Rushby's notion of a Separation Kernel [18] and Wittbold and Johnson's Nondeducibility on Strategies [17]. Separability is also stronger than Noninference and Generalized Noninference. In fact, its combination of strength and simplicity make it close to being an ideal security property for nondeterministic systems, although it is limited to systems where low-level events cannot affect high-level events.[5]

A property that allows low-level events to influence high level events can be obtained by generalizing the function *interleave* in the same way that the class of input purges generalizes the function *purge*. Say that $f : trace(\hat{\mathbf{\Sigma}}) \times trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$ is an *input interleaving* if and only if $f(t1, t2) = t$ implies that $highin(t) = highin(t1)$, $lowin(t) = lowin(t2)$, and $lowout(t) = lowout(t2)$. Generalized Noninterference, originally due to McCullough [8], is the property that a system possesses if it is closed under some input interleaving.[6]

What all of these security properties have in common is that each is a closure property with respect to some function that takes two traces and interleaves them to form a third trace. This observation motivates the following definition:

Definition 2.3 (Selective Interleaving Functions) Let Σ be the state space $\{\langle\langle in_1, ..., in_m\rangle, \langle out_1, ..., out_n\rangle\rangle \mid in_i \in I_i \wedge out_i \in O_i\}$, let $i \in \{0,1,2\}^m$, and let $j \in \{0,1,2\}^n$. A function $f : trace(\Sigma) \times trace(\Sigma) \rightarrow trace(\Sigma)$ is a *selective interleaving function of type* $F_{i,j}$ if and only if $f(t1, t2) = t$ implies that for all x such that $i[x] = 1 :\ in[x](t) = in[x](t1)$, for all x such that $i[x] = 2 :\ in[x](t) = in[x](t2)$, for all x such that $j[x] = 1 :\ out[x](t) = out[x](t1)$, and for all x such that $j[x] = 2 :\ out[x](t) = out[x](t2)$. □

Intuitively, a selective interleaving function of type $F_{i,j}$ takes its two argument traces and forms a new trace that agrees with the first argument trace with respect to input (output) channels such that $i[x]$ ($j[x]$) is equal to 1 and with the second argument trace with respect to input (output) channels such that $i[x]$ ($j[x]$) is equal to 2. Distinct selective interleaving functions of type $F_{i,j}$ differ on what they assign to input channels such that $i[x] = 0$ and output channels such that $j[x] = 0$. Hence, given i and j such that for no x does $i[x] = 0$ or $j[x] = 0$, $F_{i,j}$ contains exactly one member.

For example, Separability's function *interleave* is the single selective interleaving function of type $F_{\langle 1^H:2^L\rangle,\langle 1^H:2^L\rangle} : trace(\hat{\mathbf{\Sigma}}) \times trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$, and Noninference's *purge* is the one argument function one obtains by restricting Separability's *interleave* to the domain $\{\langle\langle\lambda^{H+L}, \lambda^{H+L}\rangle, ...\rangle\} \times trace(\hat{\mathbf{\Sigma}})$. Generalized Noninterference's input interleavings are the class of selective interleaving functions of type $F_{\langle 1^H:2^L\rangle,\langle 0^H:2^L\rangle} : trace(\hat{\mathbf{\Sigma}}) \times trace(\hat{\mathbf{\Sigma}}) \rightarrow trace(\hat{\mathbf{\Sigma}})$, and Generalized Noninference's input purges are Generalized Noninterference's input interleavings restricted to the domain $\{\langle\langle\lambda^{H+L}, \lambda^{H+L}\rangle, ...\rangle\} \times trace(\hat{\mathbf{\Sigma}})$.

From this it is clear that for any system that contains $\langle\langle\lambda^{H+L}, \lambda^{H+L}\rangle, ...\rangle$, Separability is strictly stronger than Noninference and Generalized Noninterference is strictly stronger than Generalized Noninference. Further, since any selective interleaving function of type $F_{\langle 1^H:2^L\rangle,\langle 1^H:2^L\rangle}$ is also of type $F_{\langle 1^H:2^L\rangle,\langle 0^H:2^L\rangle}$, we see that Separability is strictly stronger than Generalized Noninterference and that Noninference is strictly stronger than Generalized Noninference. Hence, Separability is the strongest of our properties, and Generalized Noninference is the weakest. Generalized Noninterference and Noninference fall in between these two, but are not comparable with each other. (See *Figure 1*.)

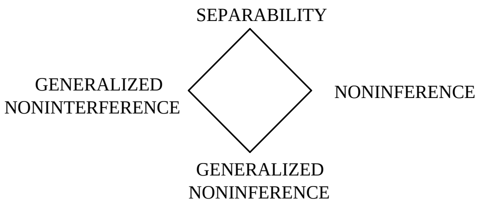

Figure 1: Partial Ordering of Possibilistic Security Models.

It is obvious that closure under a class of selective interleaving functions is not generally preserved by refinement. However, we shall see some conditions under which it is preserved in *Section 3.2*. It is also obvious that every system is closed under the selective interleaving function of type $F_{\langle 1,...,1\rangle,\langle 1,...,1\rangle}$ and the selective interleaving function of type $F_{\langle 2,...,2\rangle,\langle 2,...,2\rangle}$ and that given a trace space Σ, only the trace sets $\{\}$ and $trace(\Sigma)$ are closed under all selective interleaving functions of type $F_{\langle 0,...,0\rangle,\langle 0,...,0\rangle}$. The following theorems are also worth noting. The first shows that if a trace set is closed under one selective interleaving function, then it is closed under, at least, one other. The second shows that the identity system, $\hat{\mathbf{I}}$, is closed under a variety of selective interleaving functions.

[5] This limitation is not as stringent as it may first appear since high-level users can be allowed to read low-level input and output channels. However, it prevents a system, e.g., from recording low-level events in an audit file that is to be sent out on a high-level channel.

[6] This version of Generalized Noninterference is weaker than McCullough's by not requiring that high-level output can be altered only at a point after which high-level input has been altered. However, this difference does not affect any of the composition results that follow, and it simplifies the presentation.

Theorem 2.4 Given $s = \langle s_1, ..., s_n\rangle \in \{0, 1, 2\}^n$, let s' denote $\langle s_1{}', ..., s_n{}'\rangle$ where $0' = 0$, $1' = 2$, and $2' = 1$. Given any state space Σ and any trace set $\sigma \subseteq trace(\Sigma)$, if σ is closed under some selective interleaving function f of type $F_{i,j}$ then it is closed under some selective interleaving function f' of type $F_{i',j'}$.

Proof: For all x such that $i[x] = 0$: let $in(f'(s1, s2))[x] = in(f(s2, s1))[x]$ and for all x such that $j[x] = 0$: let $out(f'(s1, s2))[x] = out(f(s2, s1))[x]$. Note that $f'(s1, s2) = f(s2, s1)$. Hence f' is obviously a selective interleaving function of type $F_{i',j'}$. Since σ is closed under f, it is also closed under f'. □

Theorem 2.5 (Identity Theorem) For each $x \in \{1, 2\}$, the identity system, $\mathbf{\hat{I}}$, is closed under the selective interleaving function of type $F_{\langle x\rangle,\langle x\rangle}$. It is also closed under, at least, one selective interleaving function of type $F_{\langle x\rangle,\langle 0\rangle}$, at least, one selective interleaving function of type $F_{\langle 0\rangle,\langle x\rangle}$, and at least, two selective interleaving functions of type $F_{\langle 0\rangle,\langle 0\rangle}$.

Proof: The case for $F_{\langle x\rangle,\langle x\rangle}$ is obvious. For a selective interleaving function f of type $F_{\langle x\rangle,\langle 0\rangle}$ or of type $F_{\langle 0\rangle,\langle x\rangle}$, consider the selective interleaving function $f(s_1, s_2) = s_x$. For a selective interleaving function of type $F_{\langle 0\rangle,\langle 0\rangle}$, consider the selective interleaving function f^1 such that $f^1(s1, s2) = s1$ and the selective interleaving function f^2 such that $f^2(s1, s2) = s2$. □

3 System Composition

In this section we consider the composition of systems. We first consider external composition constructs, i.e. constructs used to compose a network of systems from individual systems. We then consider internal composition constructs, i.e., constructs used to compose and refine policies within one system.

3.1 External Composition Constructs

In this section we define three external composition constructs: product, cascade, and feedback. We examine the extent to which a system's closure properties with respect to classes of selective interleaving functions are preserved by each construct and show that product and feedback are sufficient for performing general composition.[7]. Our reason for separating cascade from feedback is to examine the behavior of confidentiality properties under different composition constructs. Feedback is not always necessary, and as we shall see in *Section 4*, it should be avoided whenever possible. Hence, it is useful to know how confidentiality properties behave in compositions where feedback

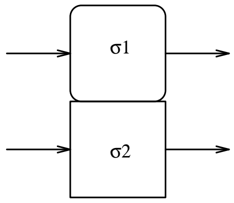

Figure 2: Product of $\sigma 1$ and $\sigma 2$

is not used.

3.1.1 Product

We begin by considering the product of two systems, i.e., the composition where two systems $\sigma 1 \subseteq \Sigma 1$ and $\sigma 2 \subseteq \Sigma 2$ are simply regarded as a single system $\sigma \subseteq \Sigma$. (See *Figure 2* .)

Definition 3.1 (Product)
Let $\Sigma 1$ and $\Sigma 2$ be any two state spaces of the form $\{\langle\langle in^1_1, ..., in^1_j\rangle, \langle out^1_1, ..., out^1_k\rangle\rangle |\ in^1_i \in I^1_i \wedge out^1_i \in O^1_i\}$ and $\{\langle\langle in^2_1, ..., in^2_m\rangle, \langle out^2_1, ..., out^2_n\rangle\rangle |\ in^2_i \in I^2_i \wedge out^2_i \in O^2_i\}$, respectively. Given any two trace sets $\sigma 1 \subseteq \Sigma 1$ and $\sigma 2 \subseteq \Sigma 2$, $\sigma 1 \times \sigma 2$ is the trace set

$$\begin{aligned} \sigma \;=\; \{s |\ & (\exists s1 \in \sigma 1)(\exists s2 \in \sigma 2) \\ & (in[1...j](s) = in(s1) \wedge \\ & in[(j+1)...(j+m)](s) = in(s2) \wedge \\ & out[1...k](s) = out(s1) \wedge \\ & out[(k+1)...(k+n)](s) = out(s2)\}. \end{aligned}$$

σ is called the *product* of $\sigma 1$ and $\sigma 2$. □

Theorem 3.2 (Composition Theorem for Products) Let $\sigma = \sigma 1 \times \sigma 2$. Then $\sigma 1$ is closed under some selective interleaving function $f^1$of type F_{i_1,j_1} and $\sigma 2$ is closed under some selective interleaving function f^2 of type F_{i_2,j_2} if and only if σ is closed under some selective interleaving function f of type $F_{\langle i_1:i_2\rangle,\langle j_1:j_2\rangle}$.

Proof: For any $s \in \sigma$ and $t \in \sigma$, let $s_{\sigma 1}$ be that part of s that is in $\sigma 1$ and $s_{\sigma 2}$ be that part of s that is in $\sigma 2$, and let $t_{\sigma 1}$ be that part of t that is in $\sigma 1$ and $t_{\sigma 2}$ be that part of t that is in $\sigma 2$. Going from left to right assume $f^1(s_{\sigma 1}, t_{\sigma 1}) = u_1$ and $f^2(s_{\sigma 2}, t_{\sigma 2}) = u_2$. We can then let

$$\begin{aligned} f(s, t) \;=\; & \langle\langle\langle in(u_1)[1] : in(u_2)[1]\rangle, \\ & \quad \langle out(u_1)[1] : out(u_2)[1]\rangle\rangle, \\ & \langle\langle in(u_1)[2] : in(u_2)[2]\rangle, \\ & \quad \langle out(u_1)[2] : out(u_2)[2]\rangle\rangle, ...\rangle, \end{aligned}$$

and we are done. Going from right to left, assume $s \in \sigma 1$ and $t \in \sigma 1$. Pick some arbitrary trace $r \in \sigma 2$, and let

[7] This has also been noted by Millen, who attributes it to Rushby, although Millen's construction differs from ours [15]

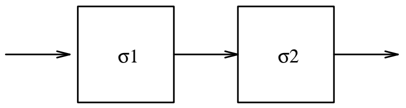

Figure 3: Cascade of $\sigma 1$ and $\sigma 2$

$$\begin{aligned} u \;=\; & \langle\langle\langle in(s)[1] : in(r)[1]\rangle, \\ & \quad \langle out(s)[1] : out(r)[1]\rangle\rangle, \\ & \langle\langle in(s)[2] : in(r)[2]\rangle, \\ & \quad \langle out(s)[2] : out(r)[2]\rangle\rangle, ...\rangle, \end{aligned}$$

and

$$\begin{aligned} v \;=\; & \langle\langle\langle in(t)[1] : in(r)[1]\rangle, \\ & \quad \langle out(t)[1] : out(r)[1]\rangle\rangle, \\ & \langle\langle in(t)[2] : in(r)[2]\rangle, \\ & \quad \langle out(t)[2] : out(r)[2]\rangle\rangle, ...\rangle. \end{aligned}$$

Letting $w = f(u, v)$, we can then let

$$\begin{aligned} f^1(s,t) \;=\; & \langle\langle in[1...j](w)[1], out[1...k](w)[1]\rangle, \\ & \langle in[1...j](w)[2], out[1...k](w)[2]\rangle, ...\rangle. \end{aligned}$$

The proof for $s \in \sigma 2$ and $t \in \sigma 2$ is analogous, and we are done. □

Corollary 3.3 Let σ be closed under some selective interleaving function of type $F_{i,j}$, and let $x \in \{0,1,2\}$ and $y \in \{0,1,2\}$ be such that either $x = 0$, $y = 0$ or $x = y$. Then $\sigma \times \hat{\mathbf{I}}$ is closed under, at least, one selective interleaving function of type $F_{\langle i:\langle x\rangle\rangle,\langle j:\langle y\rangle\rangle}$, and $\hat{\mathbf{I}} \times \sigma$ is closed under, at least, one selective interleaving function if type $F_{\langle\langle x\rangle:i\rangle,\langle\langle y\rangle:j\rangle}$. □

Proof: Use the *Identity Theorem* with the *Composition Theorem for Products.* □

3.1.2 Cascade

A more interesting type of system composition is *cascading.* (See *Figure 3* .) Cascades are formed by taking two systems $\sigma 1$ and $\sigma 2$ and passing $\sigma 1$'s output as input to $\sigma 2$. Although we assume that $\sigma 1$'s output meets any environment restrictions assumed by $\sigma 2$'s input, i.e., that $\sigma 1$'s output is acceptable input for $\sigma 2$, this assumption is used only in *Corollary 3.7.* Its purpose is to guarantee that if we the place the cascade of $\sigma 1$ and $\sigma 2$ into an environment that satisfies the input restrictions of $\sigma 1$, the resulting system will be well-behaved.

Definition 3.4 (Cascade) Let $\Sigma 1$ and $\Sigma 2$ be state spaces of the form

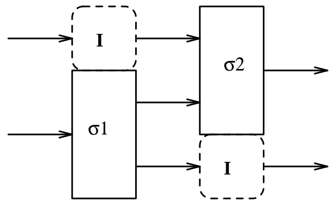

Figure 4: Using $\hat{\mathbf{I}}$ and the cascade construction to form a general cascade of $\sigma 1$ and $\sigma 2$

$\{\langle\langle in_1, ..., in_k\rangle, \langle out_1, ..., out_m\rangle\rangle |\ in_i \in I_i^1 \wedge out_i \in O_i^1\}$ and $\{\langle\langle in_1, ..., in_m\rangle, \langle out_1, ..., out_n\rangle\rangle |\ in_i \in I_i^2 \wedge out_i \in O_i^2\}$, respectively, such that $O_i^1 \subseteq I_i^2$. Given two trace sets $\sigma 1 \subseteq \Sigma 1$ and $\sigma 2 \subseteq \Sigma 2$ where for every trace $s1 \in \sigma 1$ there is a trace $s2 \in \sigma 2$ such that $out(s1) = in(s2)$, $\sigma = \sigma 1 \circ \sigma 2$ is the trace set

$$\begin{aligned} \sigma \;=\; & \{s |\ (\exists s1 \in \sigma 1)(\exists s2 \in \sigma 2)(in(s) = in(s1) \wedge \\ & \quad out(s1) = in(s2) \wedge out(s2) = out(s)\}. \end{aligned}$$

σ is called the *cascade* of $\sigma 1$ and $\sigma 2$. □

Our definition of *cascade* assumes that $\sigma 1$ has the same number of output channels as $\sigma 2$ has input channels with all of $\sigma 1$'s output going into $\sigma 2$ as input and all of $\sigma 2$'s input coming from $\sigma 1$. However, this assumption is not necessary. We can use the *Composition Theorem for Products* to append the identity system, $\hat{\mathbf{I}}$, to $\sigma 1$ so that the environment can provide input to $\sigma 2$ (*via* $\hat{\mathbf{I}}$) and to append $\sigma 2$ to $\hat{\mathbf{I}}$ so that $\sigma 1$ can provide output to the environment (also *via* $\hat{\mathbf{I}}$). We call $(\sigma 1 \times \hat{\mathbf{I}}) \circ (\hat{\mathbf{I}} \times \sigma 2)$ the *general cascade* of $\sigma 1$ and $\sigma 2$. (See *Figure 4* .) By Corollary 3.3 if $\sigma 1 \circ \sigma 2$ is closed under some selective interleaving function of type $F_{i,j}$ then the general cascade of $\sigma 1$ and $\sigma 2$ is closed under an analogous selective interleaving function, unless $i = \langle 1, ..., 1\rangle$ and $j = \langle 2, ..., 2\rangle$ or *vice versa.*

Theorem 3.5 (Composition Theorem for Cascades) Consider any two trace sets $\sigma 1$ and $\sigma 2$ as described in *Definition 3.4*, closed under selective interleaving functions f^1 of type F_{i_1,j_1} and f^2 of type F_{i_2,j_2} respectively. For any trace $\alpha \in \sigma$, let $\alpha_{\sigma 1}$ be a trace in $\sigma 1$ and $\alpha_{\sigma 2}$ be a trace in $\sigma 2$ such that $in(\alpha) = in(\alpha_{\sigma 1})$, $out(\alpha_{\sigma 1}) = in(\alpha_{\sigma 2})$, and $out(\alpha_{\sigma 2}) = out(\alpha)$. (Note that $\alpha_{\sigma 1}$ and $\alpha_{\sigma 2}$ exist by the definition of cascade.) Assume that for every s and t in σ, $f^1(s_{\sigma 1}, t_{\sigma 1}) = u_1$ implies that there is a trace $u_2 \in \sigma 2$ such that (1) $out(u_1) = in(u_2)$ and (2) for all x such that $j_2[x] \neq 0$: $out[x](u_2) = out[x](f^2(s_{\sigma 2}, t_{\sigma 2}))$. Then the function f, such that $in(f(s,t)) = in(u_1)$ and $out(f(s,t)) = out(u_2)$, is a selective interleaving function of type F_{i_1,j_2} and σ is closed under f.

Proof: For any s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, $t_{\sigma 2}$, u_1, u_2, and f be as described in the statement of the theorem. Also, let v be that sequence such that

$in(v) = u_1$ and $out(v) = u_2$. Since by the assumptions of the theorem $out(u_1) = in(u_2)$, we know that $v \in \sigma$. Hence, σ is closed under f. We shall show that f is a selective interleaving function of type F_{i_1,j_2}. To this end, note that for all x such that $i_1[x] = 1$: $in[x](v) = in[x](u_1) = in[x](s_{\sigma 1}) = in[x](s)$ and that for all x such that $i_1[x] = 2$: $in[x](v) = in[x](u_1) = in[x](t_{\sigma 1}) = in[x](t)$. Similarly, note that for all x such that $j_2[x] = 1$: $out[x](v) = out[x](u_2) = out[x](s_{\sigma 2}) = out[x](s)$ and that for all x such that $j_2[x] = 2$: $out[x](v) = out[x](u_2) = out[x](t_{\sigma 2}) = out[x](t)$. Hence, f meets all the conditions necessary to be a selective interleaving function of type F_{i_1,j_2}, and we are done. □

As an application of the *Composition Theorem for Cascades*, consider two systems $\sigma 1 \subseteq trace(\hat{\mathbf{\Sigma}})$ and $\sigma 2 \subseteq trace(\hat{\mathbf{\Sigma}})$ such that

$$\begin{aligned} \sigma 1 &= \{s \mid lowout(s) = lowin(s) \wedge \\ &\quad (i)(highout(s)[i] = highin(s)[i] + lowin(s)[i])\} \end{aligned}$$

and

$$\begin{aligned} \sigma 2 &= \{s \mid lowout(s) = lowin(s) \wedge \\ &\quad (i)(highout(s)[i] = highin(s)[i] \times lowin(s)[i])\}. \end{aligned}$$

Note that $\sigma 1$ is closed under f^1 of type $F_{\langle 1,2\rangle,\langle 0,2\rangle}$ where

$$\begin{aligned} f^1(s,t)[i] &= \langle\langle highin(s)[i], lowin(t)[i]\rangle, \\ &\quad \langle highin(s)[i] + lowin(t)[i], lowout(t)[i]\rangle\rangle \end{aligned}$$

and $\sigma 2$ is closed under f^2 of type $F_{\langle 1,2\rangle,\langle 0,2\rangle}$ where

$$\begin{aligned} f^2(s,t)[i] &= \langle\langle highin(s)[i], lowin(t)[i]\rangle, \\ &\quad \langle highin(s)[i] \times lowin(t)[i], lowout(t)[i]\rangle\rangle. \end{aligned}$$

Hence, both $\sigma 1$ and $\sigma 2$ satisfy Generalized Noninterference. By the *Composition Theorem for Cascades*, σ is closed under f of type $F_{\langle 1,2\rangle,\langle 0,2\rangle}$, where

$$\begin{aligned} f(s,t)[i] &= \langle\langle highin(s)[i], lowin(t)[i]\rangle, \\ &\quad \langle(highin(s)[i] + lowin(t)[i]) \times lowin(t)[i], \\ &\qquad lowout(t)[i]\rangle\rangle. \end{aligned}$$

Hence, σ satisfies Generalized Noninterference as well.

Although the *Composition Theorem for Cascades* is very general, it is sometimes difficult to apply since its application depends upon knowledge of system functionality to determine whether u_2 exists in $\sigma 2$. A simpler tool, which depends solely on the types of the relevant selective interleaving functions, is the following:

Corollary 3.6 Let σ, $\sigma 1$, $\sigma 2$, f^1, f^2, F_{i_1,j_1}, and F_{i_2,j_2} be as described in the *Composition Theorem for Cascades*. Given any s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, $t_{\sigma 2}$, and u_1 also be as described in that theorem. If for all $1 \leq x \leq m$: $j_1[x] = i_2[x] \neq 0$, then there is a selective interleaving function f of type F_{i_1,j_2} such that $in(f(s,t)) = in(u_1)$, $out(f(s,t)) = out(f^2(s_{\sigma 2}, t_{\sigma 2}))$, and σ is closed under f. □

Proof: Note that the restrictions on j_1 and i_2 imply that $f^2(s_{\sigma 2}, t_{\sigma 2})$ meets the conditions on u_2 required by the *Composition Theorem for Cascades*. □

As an application of *Corollary 3.6*, consider any trace sets $\sigma \subseteq trace(\hat{\mathbf{\Sigma}})$ and $\sigma 2 \subseteq trace(\hat{\mathbf{\Sigma}})$ such that $\sigma = \sigma 1 \circ \sigma 2$ is defined. Our corollary tells us the following facts:

- If $\sigma 1$ and $\sigma 2$ satisfy Separability, then so does σ.
- If $\sigma 1$ and $\sigma 2$ satisfy Noninference, then so does σ.
- If one of $\{\sigma 1, \sigma 2\}$ satisfies Noninference and the other satisfies Separability, then σ satisfies Noninference if $\langle\langle \lambda^{H+L}, \lambda^{H+L}\rangle, \ldots\rangle \in \sigma$.
- If $\sigma 1$ satisfies Separability and $\sigma 2$ satisfies Generalized Noninterference, then σ satisfies Generalized Noninterference.
- If $\sigma 1$ satisfies Noninference and $\sigma 2$ satisfies Generalized Noninference, then σ satisfies Generalized Noninference.
- If $\sigma 1$ satisfies Separability and $\sigma 2$ satisfies Generalized Noninference, then σ satisfies Generalized Noninference if $\langle\langle \lambda^{H+L}, \lambda^{H+L}\rangle, \ldots\rangle \in \sigma$.

Corollary 3.6 requires that f^1 and f^2 must agree and be fully specified with respect to interface channels, i.e., that for all $x : i_2[x] = j_1[x] \neq 0$.[8] As a consequence, although the corollary tells us about compositions where $\sigma 1$ satisfies Separability or Noninference, it tells us nothing about compositions where $\sigma 1$ satisfies Generalized Noninference or Generalized Noninterference. For such compositions we need the following:

Corollary 3.7 Let σ, $\sigma 1$, $\sigma 2$, f^1, f^2, F_{i_1,j_1}, and F_{i_2,j_2} be as described in the *Composition Theorem for Cascades* and assume that for no x does $i_2[x] = 0$. If either

$$(1) \quad ((1 \leq x \leq m \wedge j_1[x] \neq i_2[x]) \rightarrow i_2[x] = 1) \wedge (1 \leq x \leq n \rightarrow j_2[x] \neq 1)$$

or

$$(2) \quad ((1 \leq x \leq m \wedge j_1[x] \neq i_2[x]) \rightarrow i_2[x] = 2) \wedge (1 \leq x \leq n \rightarrow j_2[x] \neq 2),$$

[8] Although by *Theorem 2.4*, the corollary also applies to cases where $i_2[x] = j_1{}'[x]$. A similar observation applies to all our theorems and corollaries.

then there is a selective interleaving function f of type F_{i_1,j_2}, such that σ is closed under f. $\Box$

Proof: For any s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, $t_{\sigma 2}$, and u_1 be as in the proof of the *Composition Theorem for Cascades.* Note that although u_1 satisfies the conditions necessary for it to serve as the input part of $f(s,t)$, this case differs from the case of the *Corollary 3.6* in that we cannot use $f^2(s_{\sigma 2}, t_{\sigma 2})$ as the output part of the trace since we cannot guarantee that $out(u_1) = in(f^2(s_{\sigma 2}, t_{\sigma 2}))$. However, by the interface requirement in the definition of cascade we know that there is some trace $u^* \in \sigma 2$ such that $out(u_1) = in(u^*)$. Assume condition (1) of the theorem holds and let $u_2 = f^2(u^*, t_{\sigma 2})$. Note that for all $1 \leq x \leq m$: if $i_2[x] = 1$ then $in[x](u_2) = in[x](u^*) = out[x](u_1)$ by construction of u^*, and if $i_2[x] = 2$ then $in[x](u_2) = in[x](t_{\sigma 2}) = out[x](u_1)$ by the relationship between $t_{\sigma 1}$ and $t_{\sigma 2}$. Also, note that for all $1 \leq x \leq n$ such that $j[x] = 2$: $out[x](u_2) = out[x](t_{\sigma 2}) = out[x](t)$. Since j_2 has no 1's, u_2 fulfills all the conditions required by the *Composition Theorem for Cascades.* Condition (2) follows by an analogous argument where $u_2 = f^2(s_{\sigma 2}, u^*)$, and we are done. $\Box$

As an application of our new corollary consider any trace sets $\sigma \subseteq trace(\hat{\mathbf{\Sigma}})$ and $\sigma 2 \subseteq trace(\hat{\mathbf{\Sigma}})$ such that $\sigma = \sigma 1 \circ \sigma 2$ is defined. Our theorem tells us the following new facts:

- If $\sigma 1$ and $\sigma 2$ satisfy Generalized Noninterference, then so does σ.
- If $\sigma 1$ and $\sigma 2$ satisfy Generalized Noninference, then so does σ.
- If one of $\{\sigma 1, \sigma 2\}$ satisfies Generalized Noninference and the other satisfies Generalized Noninterference, then σ satisfies Generalized Noninference if $\langle\langle \lambda^{H+L}, \lambda^{H+L}\rangle, ...\rangle \in \sigma$.

These two facts support the following, rather interesting, observation about cascades: a possiblistic security property seems to be preserved by being cascaded with itself or with any property that is stronger than it.

3.1.3 Feedback

Another type of composition consists of a system $\sigma 1$ serving as a front end to a system $\sigma 2$ or, equivalently, $\sigma 2$ serving as a back end to $\sigma 1$. The essential element of this connection is that $\sigma 2$ provides feedback to $\sigma 1$. (See *Figure 5* .) In this case when a user provides input to $\sigma 1$ (for example, at time 1 according to the user's and $\sigma 1$'s local clocks), the output generated by this input is taken as input by $\sigma 2$ (also at time 1 by $\sigma 2$'s local clock). This input to $\sigma 2$ generates output which is read as new input by $\sigma 1$ (at time 1 of $\sigma 2$'s local clock, but now time 2 of $\sigma 1$'s local clock). The user then receives from $\sigma 1$ the output that is generated in

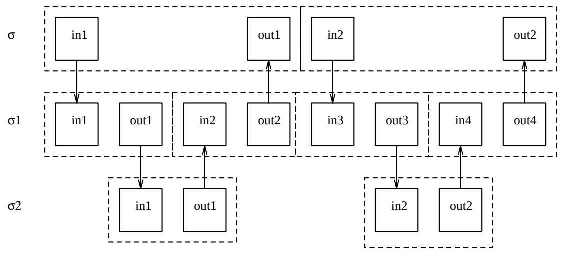

Figure 5: σ as the Feedback of $\sigma 1$ and $\sigma 2$

response to the input from $\sigma 2$ (at time 2 of $\sigma 1$'s local clock, but still time 1 of the user's local clock). The user then provides the next input to $\sigma 1$ (time 2 by the user's clock, but now time 3 by $\sigma 1$'s local clock). The process continues with the user providing $\sigma 1$ with its odd inputs at $\sigma 1$ local time τ (where τ is odd) and user local time $(\tau+1)/2$ and receiving $\sigma 1$'s even outputs at $\sigma 1$ local time τ (where τ is even) and user local time $\tau/2$. In the meantime, $\sigma 1$ sends its odd outputs to $\sigma 2$ at $\sigma 1$ local time τ (where τ is odd) and $\sigma 2$ local time $(\tau+1)/2$ and receives its even inputs from $\sigma 2$ at $\sigma 1$ local time τ (where τ is even) and $\sigma 2$ local time $\tau/2$.

As in cascading, we assume that $\sigma 1$'s (odd) output meets any environment restrictions assumed by $\sigma 2$'s input. We also assume that the output of those traces of $\sigma 2$ that take input from $\sigma 1$ meet any environment restrictions assumed by $\sigma 1$'s (even) input. Although these assumptions are not necessary for the proofs presented in this section, they will reappear in *Section 4* when we discuss the possibility of a feedback analogue for *Corollary 3.7.* As in the definition of *cascade*, their purpose is to guarantee that the feedback of $\sigma 1$ and $\sigma 2$ will be well-behaved if placed in an environment that satisfies $\sigma 1$'s (odd) input restrictions.

To formalize these interface assumptions, we cannot simply require that for every $s1 \in \sigma 1$ there is a trace $s2 \in \sigma 2$ such that for all odd τ : $out(s1)[\tau] = in(s2)[(\tau+1)/2]$ and for all even τ : $in(s1)[\tau] = out(s2)[\tau/2]$ since, in general, such a requirement is too strong. For example, although it is reasonable to require that for every trace $s1 \in \sigma 1$ there is some trace $s2 \in \sigma 2$ such that $out(s1)[1] = in(s2)[1]$ and to require that there is some trace $s1^* \in \sigma 1$ such that $s1^*[1] = s1[1] \wedge in(s1^*)[2] = out(s2)[1]$, we cannot guarantee that $s1^* = s1$ since $in(s1)[2]$ may not be a possible output for $\sigma 2$. What we need to say is that if two traces $s1 \in \sigma 1$ and $s2 \in \sigma 2$ have interfaced correctly up to $\sigma 2$ local time τ, then each trace has a "continuation" that will interface correctly at $\sigma 2$ local time τ.

Definition 3.8 (Interface Condition for Feedback) Let $\Sigma 1$ and $\Sigma 2$ be state spaces of the form $\{\langle\langle in_1, ..., in_n\rangle, \langle out_1, ..., out_m\rangle\rangle |\ in_i \in I^1_i \wedge out_i \in O^1_i\}$ and $\{\langle\langle in_1, ..., in_m\rangle, \langle out_1, ..., out_n\rangle\rangle |\ in_i \in I^2_i \wedge out_i \in O^2_i\}$, respectively, such that for all $1 \leq i \leq m$: $O^1_i \subseteq I^2_i$ and for all $1 \leq i \leq n$: $O^2_i \subseteq I^1_i$. For any trace $s1 \in \sigma 1 \subseteq trace(\Sigma 1)$ and $s2 \in \sigma 2 \subseteq trace(\Sigma 2)$ let the relation $downconnect(1, s1, s2) =_d$

$out(s1)[1] = in(s2)[1]$. For all $\tau > 0$: let the relation $upconnect(\tau, s1, s2) =_d (downconnect(\tau, s1, s2) \wedge out(s2)[\tau] = in(s1)[2\tau])$, and for all $\tau > 1$: let the relation $downconnect(\tau, s1, s2) =_d (upconnect(\tau - 1, s1, s2) \wedge in(s2)[\tau] = out(s1)[2\tau - 1])$. We say that $\sigma 1$ and $\sigma 2$ *meet the interface requirements for feedback* if and only if

- $(s1 \in \sigma 1)(\exists s2 \in \sigma 2) downconnect(1, s1, s2)$,

- $(\tau \geq 1)(s1 \in \sigma 1)(s2 \in \sigma 2)(\exists s^* \in \sigma 1)$
  $(downconnect(\tau, s1, s2) \rightarrow$
  $(s^*[1...(2\tau - 1)] = s1[1...(2\tau - 1)] \wedge$
  $upconnect(\tau, s^*, s2)))$,

and

- $(\tau \geq 1)(s1 \in \sigma 1)(s2 \in \sigma 2)(\exists s^* \in \sigma 2)$
  $(upconnect(\tau, s1, s2) \rightarrow$
  $downconnect(\tau + 1, s1, s^*))$.

□

Definition 3.9 (Feedback) Let $\sigma 1$ and $\sigma 2$ be as described in *Definition 3.8* so that they meet the interface condition for feedback. $\sigma = \sigma 1 \stackrel{\leftharpoonup}{\rightharpoondown} \sigma 2$ is the trace set

$$\begin{aligned} \sigma = \{ s | & \ (\exists s1 \in S1)(\exists s2 \in S2) \\ & (in(s)[\tau] = in(s1)[2\tau - 1] \wedge \\ & out(s)[\tau] = out(s1)[2\tau] \wedge \\ & in(s2)[\tau] = out(s1)[2\tau - 1] \wedge \\ & out(s2)[\tau] = in(s1)[2\tau]) \}. \end{aligned}$$

σ is called the *feedback* of $\sigma 1$ *and* $\sigma 2$. □

As in cascading, although we assume that the interface channels of $\sigma 1$ and $\sigma 2$ can be placed in one-to-one correspondence, this assumption is not necessary. Using $\hat{\mathbf{I}}$ and the feedback construction, one can form a general hook-up. (See figure 6.) We call $(\sigma 1 \times \hat{\mathbf{I}}) \stackrel{\leftharpoonup}{\rightharpoondown} (\hat{\mathbf{I}} \times \sigma 2)$ the *general composition* of $\sigma 1$ and $\sigma 2$. As in the case of general cascades, the general composition of $\sigma 1$ and $\sigma 2$ preserves all interesting closure properties that are preserved by $\sigma 1 \stackrel{\leftharpoonup}{\rightharpoondown} \sigma 2$.

Our composition theorem for feedback considers the case where $\sigma 1$ and $\sigma 2$ are intimately connected.

Theorem 3.10 (Composition Theorem for Feedback) Let σ, $\sigma 1$ and $\sigma 2$ be as described in *Definition 3.9*, and for any trace $\alpha \in \sigma$, let $\alpha_{\sigma 1}$ be a trace in $\sigma 1$ such that for all τ : $in(\alpha)[\tau] = in(\alpha_{\sigma 1})[2\tau - 1] \wedge out(\alpha)[\tau] = out(\alpha_{\sigma 1})[2\tau]$, and let $\alpha_{\sigma 2}$ be a trace in $\sigma 2$ such that for all τ : $in(\alpha_{\sigma 2})[\tau] = out(\alpha_{\sigma 1})[2\tau - 1] \wedge out(\alpha_{\sigma 2})[\tau] = in(\alpha_{\sigma 1})[2\tau]$. (Note that $\alpha_{\sigma 1}$ and $\alpha_{\sigma 2}$ exist by the definition of feedback.)

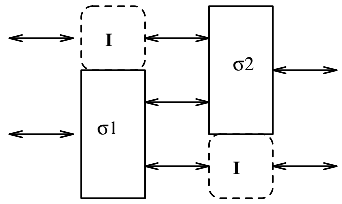

Figure 6: Using $\hat{\mathbf{I}}$ and the Feedback Construction to form a General Hook-up

Assume that $\sigma 1$ is closed under some selective interleaving function f^1 of type F_{i_1,j_1}. If for every trace s and t in σ, $f^1(s_{\sigma 1}, t_{\sigma 1}) = u_1$ implies that there is a trace $u_2 \in \sigma 2$ such that for all τ : $in(u_2)[\tau] = out(u_1)[2\tau - 1]$ and $out(u_2)[\tau] = in(u_1)[2\tau]$, then σ is closed under some selective interleaving function f of type F_{i_1,j_1} such that

$$\begin{aligned} f(s,t) &= \langle \langle in(u_1)[1], out(u_1)[2] \rangle, \\ & \quad \langle in(u_1)[3], out(u_1)[4] \rangle, \\ & \quad \langle in(u_1)[5], out(u_1)[6] \rangle, ... \rangle. \end{aligned}$$

Proof: For any traces s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, $t_{\sigma 2}$, and u_1 be as in the theorem. Note that for all x such that $i_1[x] = 1 : in[x](s)[\tau] = in[x](s_{\sigma 1})[2\tau - 1] = in[x](u_1)[2\tau - 1]$ and that for all x such that $i_1[x] = 2 : in[x](t)[\tau] = in[x](t_{\sigma 1})[2\tau - 1] = in[x](u_1)[2\tau - 1]$. Also note that for all x such that $j_1[x] = 1 : out[x](s)[\tau] = out[x](s_{\sigma 1})[2\tau] = out[x](u_1)[2\tau]$ and that for all x such that $j_1[x] = 2 : out[x](t)[\tau] = out[x](t_{\sigma 1})[2\tau] = out[x](u_1)[2\tau]$. Hence, f as defined in the theorem is a selective interleaving function of type F_{i_1,j_1}. All that is left is to show that $f(s,t) \in \sigma$. To do this we must show that there is a trace $u_2 \in \sigma 2$ that correctly interfaces with u_1. However, the existence of u_2 is guaranteed by the assumptions of our theorem, and we are done. □

As an example of the *Composition Theorem for Feedback*, consider the system $\sigma \subseteq traces(\hat{\mathbf{\Sigma}})$ such that $lowout_i = lowin_i$ and for any high-level input, $highout_i$ randomly ranges over every value in its domain. In other words, σ echoes low-level input and produces random high-level output given any high-level input. σ satisfies Generalized Noninterference. By the *First Composition Theorem for Feedback*, $\sigma \stackrel{\leftharpoonup}{\rightharpoondown} \sigma$ does as well.

As in the case of the *Composition Theorem for Cascades*, the *Composition Theorem for Feedback* is general, but hard to apply since it requires knowledge of system functionality to determine whether u_2 exists in $\sigma 2$. The following corollary provides a simpler tool:

Corollary 3.11 Let σ, $\sigma 1$, $\sigma 2$, f^1, and F_{i_1,j_1} be as described in the *Composition Theorem for Feedback*, and given any s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, $t_{\sigma 2}$, and u_1 also be as described in that theorem. Assume that $\sigma 2$ is closed under a selective interleaving function f^2

of type F_{i_2,j_2}. If $i_1 = j_2$, $i_2 = j_1$, and there is no x such that $i_1[x] = 0$ or $i_2[x] = 0$, then there is a selective interleaving function f of type F_{i_1,j_1}, such that σ is closed under f and

$$\begin{aligned} f(s,t) &= \langle\langle in(u_1)[1], out(u_1)[2]\rangle, \\ &\quad \langle in(u_1)[3], out(u_1)[4]\rangle, \\ &\quad \langle in(u_1)[5], out(u_1)[6]\rangle, ...\rangle. \end{aligned}$$

□

Proof: The corollary follows from the *Composition Theorem for Feedback* if we can show that there is some $u_2 \in \sigma 2$ that interfaces correctly with u_1. To see that there is such a trace in $\sigma 2$, consider $u_2 = f^2(s_{\sigma 2}, t_{\sigma 2})$. Since $i_1 = j_2$ and $i_2 = j_1$, the fact that neither i_1 nor i_2 contain any 0's implies that u_2 satisfies the conditions required by the *Composition Theorem for Feedback*, and we are done. □

Consider any trace sets $\sigma \subseteq trace(\hat{\mathbf{\Sigma}})$ and $\sigma 2 \subseteq trace(\hat{\mathbf{\Sigma}})$ such that $\sigma = \sigma 1 \overset{\leftarrow}{\rightarrow} \sigma 2$ is defined. The following facts are consequences of our corollary:

- If $\sigma 1$ and $\sigma 2$ satisfy Separability then so does σ.
- If $\sigma 1$ and $\sigma 2$ satisfy Noninference, then so does σ.
- If one of $\{\sigma 1, \sigma 2\}$ satisfies Separability and the other satisfies Noninference, then σ satisfies Noninference if $\langle\langle\lambda^{H+L}, \lambda^{H+L}\rangle, ...\rangle \in \sigma$.

Note that *Corollary 3.11* for feedback is the analogue of *Corollary 3.6* for cascade. There is no feedback analogue of *Corollary 3.7*. We shall examine the reason for this in *Section 4*.

3.2 Internal Composition Constructs

We now consider three elementary types of internal composition: $\sigma 1 \cup \sigma 2$, $\sigma 1 \cap \sigma 2$, and $\sigma 1 - \sigma 2$. The set consisting of these three composition constructions, which we shall call the *set of regular composition constructions*, is analogous to the set of constructions defined for access control policies in [9]. The first construction corresponds to a system that accepts any input acceptable to $\sigma 1$ or $\sigma 2$ and behaves as the relevant system would behave. If the input is acceptable to both systems, then output could be the output of either system. The second construction accepts as input only input that is acceptable to both systems and gives as output only output that both systems could generate. The final construction accepts as input only input $\sigma 2$ would not accept. Since the latter two constructions can obviously be used to refine a property, their composition properties also tell us about secure refinements. In general, the conditions for preserving closure properties with such constructs are very restrictive. We shall gain some insight into why this is the case in the next section.

Theorem 3.12 (Composition Theorem for Set Union) Assume that for some state space Σ, trace sets $\sigma 1 \subseteq \Sigma$ and $\sigma 2 \subseteq \Sigma$ are closed under selective interleaving functions f^1 of type F_{i_1,j_1} and f^2 of type F_{i_2,j_2}, respectively, such that for all $x : i_1[x] \neq i_2[x] \rightarrow i_2[x] = 0$ and $j_1[x] \neq j_2[x] \rightarrow j_2[x] = 0$. Also assume that for each pair of traces $s1 \in \sigma 1$ and $s2 \in \sigma 2$ there is (1) either a trace $t \in \sigma 1$ such that for all x such that $i_1[x] = 2$: $in[x](t) = in[x](s2)$ and for all x such that $j_1[x] = 2$: $out[x](t) = out[x](s2)$ or a trace $t \in \sigma 2$ such that for all x such that $i_2[x] = 1$: $in[x](t) = in[x](s1)$ and for all x such that $j_2[x] = 1$: $out[x](t) = out[x](s1)$ and (2) either a trace $t \in \sigma 1$ such that for all x such that $i_1[x] = 1$: $in[x](t) = in[x](s2)$ and for all x such that $j_1[x] = 1$: $out[x](t) = out[x](s2)$ or a trace $t \in \sigma 2$ such that for all x such that $i_2[x] = 2$: $in[x](t) = in[x](s1)$ and for all x such that $j_2[x] = 2$: $out[x](t) = out[x](s1)$. Then $\sigma 1 \cup \sigma 2$ is closed under some selective interleaving function f of type F_{i_2,j_2}.

Proof: We shall define a value of $f(s1, s2)$ for each $s1$ and $s2$ in $\sigma 1 \cup \sigma 2$. If $s1$ and and $s2$ are both in $\sigma 1$, then $f(s1, s2) = f^1(s1, s2)$. If $s1$ and and $s2$ are both in $\sigma 2 - \sigma 1$, then $f(s1, s2) = f^2(s1, s2)$. If $s1 \in \sigma 1$ and $s2 \in \sigma 2$, then note that by the first assumption in the theorem there is either a trace $t \in \sigma 1$ such that for all x such that $i_1[x] = 2$: $in[x](t) = in[x](s2)$ and for all x such that $j_1[x] = 2$: $out[x](t) = out[x](s2)$ or a trace $t \in \sigma 2$ such that for all x such that $i_2[x] = 1$: $in[x](t) = in[x](s1)$ and for all x such that $j_2[x] = 1$: $out[x](t) = out[x](s1)$. Assume that the first possibility is the case. Then let $f(s1, s2) = f^1(s1, t)$. If the first possibility does not hold, then the second possibility must hold and we can let $f(s1, s2) = f^2(t, s2)$. Since by the assumptions on i_1, j_1, i_2, and j_2 any function of type F_{i_1,j_1} is also of type F_{i_2,j_2}, f as defined is of type F_{i_2,j_2}. If $s1 \in \sigma 2$ and $s2 \in \sigma 1$ then an analogous argument applies using the second assumption of the theorem, and we are done. □

Theorem 3.13 (Composition Theorem for Set Intersection) Assume that for some state space Σ, trace sets $\sigma 1 \subseteq \Sigma$ and $\sigma 2 \subseteq \Sigma$ are closed under selective interleaving functions f^1 of type F_{i_1,j_1} and f^2 of type F_{i_2,j_2}, respectively, such that for all $x : i_1[x] \neq i_2[x] \rightarrow i_2[x] = 0$ and $j_1[x] \neq j_2[x] \rightarrow j_2[x] = 0$. If for all $s1 \in \sigma 1 \cap \sigma 2$ and $s2 \in \sigma 1 \cap \sigma 2$: $f^1(s1, s2) \in \sigma 2$ or $f^2(s1, s2) \in \sigma 1$, then $\sigma = \sigma 1 \cap \sigma 2$ is closed under some selective interleaving function f of type F_{i_2,j_2}.

Proof: As in the proof of the *Composition Theorem for Set Union* any function of type F_{i_1,j_1} is also of type F_{i_2,j_2}. Consider any $s1$ and $s2$ in $\sigma 1 \cap \sigma 2$. By assumption $f^2(s1, s2) \in \sigma 2$. If $f^2(s1, s2) \in \sigma 1$ as well, then we can simply let $f(s1, s2) = f^2(s1, s2)$. Otherwise, we know that $f^1(s1, s2) \in \sigma 1 \cap \sigma 2$ by the assumptions of the theorem. In this case we can let $f(s1, s2) = f^1(s1, s2)$, and we are done. □

Theorem 3.14 (Composition Theorem for Set Subtraction) Assume trace sets $\sigma 1$ and $\sigma 2$ such that

$\sigma 1$ is closed under some selective interleaving function f of type $F_{i,j}$. Assume also that for each trace $s \in \sigma 1 \cap \sigma 2$ such that there are traces $s1 \in \sigma 1 - \sigma 2$ and $s2 \in \sigma 1 - \sigma 2$ where $f(s1, s2) = s$, there is a trace $s^* \in \sigma 1 - \sigma 2$ such that for all x such that $i[x] \neq 0$: $in[x](s^*) = in[x](s)$, and for all x such that $j[x] \neq 0$: $out[x](s^*) = out[x](s)$. Then there is a selective interleaving function f^* of type $F_{i,j}$ such that $\sigma 1 - \sigma 2$ is closed under f^*.

Proof: We can let $f^* = f$ for all arguments $(s1, s2)$ except for the case where $s1 \in \sigma 1 - \sigma 2$ and $s2 \in \sigma 1 - \sigma 2$, but $f(s1, s2) \in \sigma 1 \cap \sigma 2$, i.e., $\sigma 1 - \sigma 2$ contains $s1$ and $s2$, but not $f(s1, s2)$. However, by assumption we know that in this case there is a trace $s^* \in \sigma 1 - \sigma 2$ such that for all x such that $i[x] \neq 0$: $in[x](s^*) = in[x](f(s1, s2))$, and for all x such that $j[x] \neq 0$: $out[x](s^*) = out[x]((f(s1, s2))$. Let $f^*(s1, s2) = s^*$, and we are done. □

Corollary 3.15 (Secure Refinement) Let trace set σ be closed under selective interleaving function f of type $F_{i,j}$ and let σ^* be a refinement of σ. Then σ^* is closed under a selective interleaving function f^* of type $F_{i,j}$ if either (1) there is some $\hat{\sigma}$ such that $\sigma^* = \sigma \cap \hat{\sigma}$, and $\hat{\sigma}$ and σ meet the condition stated in *Theorem 3.13* for $\sigma 1$ and $\sigma 2$, respectively, or (2) σ and $\sigma - \sigma^*$ meet the conditions stated in *Theorem 3.14* for $\sigma 1$ and $\sigma 2$, respectively. □

Proof: Condition (1) follows directly from *Theorem 3.13*. Condition (2) follows from *Theorem 3.14* since $\sigma^* \subseteq \sigma$ implies that $\sigma - (\sigma - \sigma^*) = \sigma^*$. □

4 Discussion

Although we have considered only 2-level security policies, it should be noted that 2-argument selective interleaving functions can capture multi-level policies as well. For example, a 3-level Separability policy on a state space where level i is assigned input channel in_i and output channel out_i is the requirement that a trace set be closed under selective interleaving functions of type $F_{\langle 1,2,2\rangle,\langle 1,2,2\rangle}$, $F_{\langle 2,1,2\rangle,\langle 2,1,2\rangle}$, and $F_{\langle 2,2,1\rangle,\langle 2,2,1\rangle}$.

One benefit of our approach is the new results it has generated. We have seen several theorems about selective interleavings and about the composability of closure properties with respect to selective interleavings, which we have applied to several security properties. This has given us new facts about the relationships among these properties and about their composability with each other and with themselves. One observation we have made is that a property seems to be preserved by being cascaded with itself or with a stronger property. Another is that Separability seems to be just as composable as both Noninference, which is less secure than Separability, and Restrictiveness, which is more complicated than Separability. We have also shown that even for systems not suitable for Separability (i.e., systems where low-level users affect high-level output), we do not have to resort to Restrictiveness if we limit ourselves to certain the composition constructs of product and cascade.

Another benefit is that our approach sheds new light on familiar results. For example, although McCullough showed that Generalized Noninterference is not preserved when a system σ is composed by general composition from component systems $\sigma 1$ and $\sigma 2$, his example gives no indication whether the problem with general composition is the fact that $\sigma 1$ provides input to $\sigma 2$, the fact that $\sigma 2$ provides feedback to $\sigma 1$, or the fact that $\sigma 2$ can provide direct output to the environment. (His example does not require that the environment provide direct input to $\sigma 2$). Given our results about cascaded systems, we can see that feedback is the culprit.

To understand why, consider McCullough's example in more detail. To construct σ, McCullough considered a system $\sigma 1$ which receives arbitrary high-level input and responds with a high-level output for each input. It may also receive a low-level input of *cancel*, to which it will eventually respond by sending a low-level output of *cancel*. If when the low-level output is sent the number of high-level inputs is equal to the number of high-level outputs, the low-level output *nothing-to-cancel* may be sent as well (but it does not have to be). System $\sigma 2$ is the same as $\sigma 1$ but a low-level output of *cancel* is not sent. If when the low-level input is received the number of high-level inputs is equal to the number of high-level outputs, the low-level output $nothing-to-cancel$ may be sent as well (but it does not have to be). The system σ is composed from $\sigma 1$ and $\sigma 2$ by sending $\sigma 1$'s high-level output and low-level output of *cancel* to $\sigma 2$ as input and sending $\sigma 2$'s high-level output to $\sigma 1$ as input. System $\sigma 2$ can receive no input from the user. The output of σ is the Cartesian product of the low-level outputs of $\sigma 1$ and $\sigma 2$.

The problem with σ is that there is no corollary of the *Composition Theorem for Feedback* that corresponds to *Corollary 3.7* of the *Composition Theorem for Cascade*. The conditions a trace must meet to be in σ are too strong to support such a corollary since they require, not only that the output of some trace in $\sigma 1$ be acceptable as input to some trace in $\sigma 2$ (a condition also required by cascade), but also that the output of the latter trace be acceptable as input to the former. This second requirement severely cuts back on the number of traces a system composed *via* feedback can exhibit. Given any two legal traces s and t, possibilistic security properties require the existence of a third trace $f(s,t)$ that combines the first two. Hence, it is understandable why not many such properties are preserved by constructions that make it hard for $f(s,t)$ to exist.[9]

[9] This also explains why security does not do well under internal composition. The union construct tends to increase the number of legal traces s and t more quickly than it increases the number of traces $f(s,t)$, and both the intersection and set difference constructions decrease the number of traces $f(s,t)$.

It might seem that we could prove the necessary corollary by applying the *Interface Condition for Feedback* and the same trick used to prove *Corollary 3.7*. For example, let σ, $\sigma 1$, $\sigma 2$, f^1, f^2, F_{i_1,j_1}, and F_{i_2,j_2} be as described in the *Corollary 3.11*, and given any s and t in σ, let $s_{\sigma 1}$, $s_{\sigma 2}$, $t_{\sigma 1}$, and $t_{\sigma 2}$ also be as described in that corollary with $u_1 = f^1(s_{\sigma 1}, t_{\sigma 1})$. Assume that for no x does $i_2[x] = 0$ and that the first condition of *Corollary 3.7* holds, i.e.,

$$((1 \leq x \leq m \wedge j_1[x] \neq i_2[x]) \rightarrow i_2[x] = 1) \wedge \\ (1 \leq x \leq n \rightarrow j_2[x] \neq 1).$$

Now, even if $i_2 \neq j_1$, we know that there is a trace $v^* \in \sigma 2$ such that $in(v^*)[1] = out(u_1)[1]$. Hence, we could let v_1 be $f^2(v^*, t_{\sigma 2})$ and know that $in(v_1)[1] = out(u_1)[1]$. However, to know that $in(u_1)[2] = out(v_1)[1]$, we would have to assume that for all $1 \leq y \leq n: \ i_1[y] = j_2[y] = 2$. This would yield the result that σ is closed under a selective interleaving function of type $F_{\langle 2,\ldots,2\rangle,\langle 2,\ldots,2\rangle}$, but this result trivially holds for all trace sets. Another approach would be to construct a trace in $\sigma 1$ from u_1 to interface with v_1. Assuming that i_1, j_2, and j_1 meet the same conditions as i_2, j_1, and j_2, respectively, we could use the fact that there is some trace $w^* \in \sigma 1$ such that $w^*[1] = u_1[1]$ and $in(w^*)[2] = out(v_1)[1]$ to form $w_1 = f^1(w^*, u_1)$ to interface with v_2. We could continue in this fashion constructing two sequence of traces v_i and w_i that interface with each other at times $1, .., i$. This would prove:

$$(1) \qquad (i)(\exists w \in \sigma 1)(\exists v \in \sigma 2)(x) \\ (1 \leq x \leq i \rightarrow \\ (in(v)[x] = out(w)[2x-1] \wedge \\ out(v)[x] = in(w)[2x])).$$

However, to show that σ is closed under a selective interleaving function of the form F_{i_1,j_1}, we would have to prove:

$$(2) \qquad (\exists w \in \sigma 1)(\exists v \in \sigma 2)(i) \\ (in(v)[i] = out(w)[2i-1] \wedge \\ out(v)[i] = in(w)[2i])).$$

The first statement says that for every time i we can find two traces that interface correctly together up through i. The second statement says that we can find two traces that interface together correctly at every time i. Although the second implies the first, the first does not imply the second.

Returning to the McCullough example, note that there is a large set of traces in both $\sigma 1$ and $\sigma 2$ that accept high-level input and produce $nothing-to-cancel$. Further, for any time i, there is a pair of traces $s1 \in \sigma 1$ and $s2 \in \sigma 2$, each containing both high-level input and $nothing-to-cancel$, such that $s1$ and $s2$ interface correctly through i. However, none of these trace pairs interface correctly for all times i. Hence, a high-level input to σ rules out an otherwise acceptable low-level output of $\langle nothing-to-cancel, nothing-to-cancel\rangle$.

From an information-theoretic viewpoint, the feedback might simply be exacerbating a high-to-low channel that is already present in the two component systems since it is possible that a Trojan Horse can use $\sigma 1$ or $\sigma 2$ to pass information by altering the probability that a low-level user will see the output $nothing-to-cancel$. For example, assume that (1) if there is nothing to cancel, then the system under consideration gives the output $nothing-to-cancel$ 50% of the time, (2) the Trojan Horse can block high-level input from the user, and (3) the Trojan Horse can submit high-level inputs at such a rate that there is a nonzero probability that the system will still be processing high-level inputs when the *cancel* output is given (in the case of $\sigma 1$) or when the *cancel* input is received (in the case of $\sigma 2$). A Trojan Horse can send a 1 to the low-level user by flooding the system with high-level inputs, thereby lowering the probability that $nothing-to-cancel$ will appear as low-level output to under 50%, and it can send a 0 to the low-level user by blocking all high-level inputs, thereby assuring that the probability that $nothing-to-cancel$ will appear as low-level output is 50%.

However, the feedback might also be creating a channel where none existed before. Since the Trojan Horse can transmit information only if we assume that it can lower the probability that $nothing-to-cancel$ will appear as low-level output, we can effectively shut down the Trojan Horse in each component by speeding up the component's speed relative to the Trojan Horse's (i.e., so the component can process the Trojan Horse's high level inputs and send them on their way as high-level outputs faster than the Trojan Horse can produce them). If we can make each component very fast relative to the Trojan Horse, there will never be any high-level input that will be cancelled. In such a system, $nothing-to-cancel$ will appear as low-level output 50% of the time, independently of what the Trojan Horse does.

However, although each component system may have a high-to-low capacity of 0, σ still has a positive capacity. If the Trojan Horse blocks all high-level input, each component system has a 50% chance of producing $nothing-to-cancel$ and there is a 25% chance that both components will produce $nothing-to-cancel$. However, if the Trojan Horse floods the system, there is a 0% chance of both systems producing $nothing-to-cancel$.[10] Hence, we have connected two systems with 0 high-to-low capacity to form a composite system with positive high-to-low capacity. For example, assume that we have no control over which system will be performing high-level processing at the time high-level processing is killed. In this case the composite system's low-level output when the Trojan Horse blocks high-level input will be $\langle \lambda, \lambda\rangle$ 25% of the time, $\langle\langle nothing-to-cancel\rangle, \lambda\rangle$ 25% of the time, $\langle \lambda, \langle nothing-to-cancel\rangle\rangle$ 25% of the time,

[10] I am assuming that communication between $\sigma 1$ and $\sigma 2$ is instantaneous so that there is no chance of the high-level input being "between" systems when high-level processing stops. McCullough must make the same assumption for σ to fail to satisfy Generalized Noninterference.

and $\langle\langle nothing-to-cancel\rangle, \langle nothing-to-cancel\rangle\rangle$ 25% of the time. When the Trojan Horse submits a high-level input, low-level output will be $\langle\lambda, \lambda\rangle$ 50% of the time, $\langle\langle nothing-to-cancel\rangle, \lambda\rangle$ 25% of the time, $\langle\lambda, \langle nothing-to-cancel\rangle\rangle$ 25% of the time, and $\langle\langle nothing-to-cancel\rangle, \langle nothing-to-cancel\rangle\rangle$ 0% of the time. The resulting channel has a capacity of about .17 bits per symbol. In fact, even if we limit ourselves to looking at a single low-level channel (e.g., the output from $\sigma 1$ or the output from $\sigma 2$), there is a high-to-low channel of positive capacity in the composed system.

5 Conclusion

We have constructed a general framework for specifying and reasoning about compositions of a class of properties that fall outside of the safety/liveness domain of [2], and we have shown the framework's applicability to possibilistic security properties. The framework we have developed has allowed us to partially order several possibilistic security properties and to examine their composability. We have seen that properties do quite well when composed with themselves or with stronger properties *via* the product and cascade construction. However, survival under feedback and internal constructions (including refinement) is contingent upon particulars of system functionality. We have looked at the reason for this.

Along the way we have presented a new model, Separability, and we have shown that if we can live with its limitation that it can be applied only to systems where low-level events cannot affect high-level events, it provides a composable formulation of secrecy that is simpler than, yet just as secure as, Restrictiveness and more secure, though no more complicated than, Noninference.

The framework and theorems presented in this paper form the building blocks of a general theory of system properties and their composition, one of whose applications is security. The framework has the advantage that it fits in well with other computer science modeling frameworks, e.g., [2] and with frameworks for modeling probabilistic systems, e.g., [5]. This allows us to bring in results from general computer science and extend our results to probabilistic models in the future. By jettisoning the requirement of input totality, our framework allows us to use assumptions about system environments to simplify the analysis of embedded systems.

Acknowledgements

I wish to thank James Gray for calling my attention to the framework presented in [2, 1] and the limitations such frameworks posses when dealing with security properties. I also wish to thank Gray, Sue Landauer, Frédéric Cuppens, Judy Hemenway, Dale Johnson, Catherine Meadows, and John Rushby for their written comments on previous drafts of this paper. Useful conversations with George Dinolt and Paul Syverson increased the paper's clarity.

References

[1] Martin Abadi and Leslie Lamport, *Composing Specifications,* Technical Report 66, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, 1990.

[2] Bowen Alpern and Fred Schneider, "Defining Liveness," *Information Processing Letters*, 21(4):181-185, October 1985.

[3] Joseph Goguen and Jose Meseguer, "Security Policies and Security Models," *Proceedings of the 1982 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1982.

[4] John Graham-Cumming and J. W. Sanders, "On the Refinement of Non-interference," *Proceedings of the Computer Security Foundations Workshop IV*, IEEE Press, 1991.

[5] James Gray, "Toward a Mathematical Foundation for Information Flow Security," *Journal of Computer Security*, 1(3-4):255-294, 1992.

[6] Jeremy Jacob, "On the Derivation of Secure Components," *Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1989.

[7] Ming Li and Paul Vitányi, "Kolmogorov Complexity and its Applications," in *Handbook of Theoretical Computer Science, vol. A: Algorithms and Complexity* (ed. J. van Leeuwen), MIT Press/Elsevier, 1990.

[8] Daryl McCullough, "Specifications for Multi-Level Security and a Hook-Up Property," *Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1987.

[9] John McLean, "The Specification and Modeling of Computer Security," *Computer*, 23(1):9-16, 1990.

[10] John McLean, "Security Models and Information Flow," *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1990.

[11] John McLean, "Proving Noninterference and Functional Correctness Using Traces," *Journal of Computer Security*, 1(1):37-57, 1992.

[12] John McLean, "Models of Confidentiality: Past, Present, and Future," *Proceedings of the Computer Security Foundations Workshop VI*, IEEE Press, 1993.

[13] John McLean, "Security Models," in the *Encyclopedia of Software Engineering*, Wiley, in press.

[14] Catherine Meadows, "Using Traces Based on Procedure Calls to Reason about Composability," *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1992.

[15] Jonathan Millen, "Hookup Security for Synchronous Machines," *Proceedings of the Computer Security Foundations Workshop III*, IEEE Press, 1990.

[16] Colin O'Halloran, "A Calculus of Information Flow," *Proceedings of the European Symposium on Research in Computer Security,* Toulouse, France, 1990.

[17] J. Todd Wittbold and Dale Johnson, "Information Flow in Nondeterministic Systems," *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1990.

[18] John Rushby, "Design and Verification of Secure Systems," *Proceedings of the Eighth Symposium on Operating System Principles*, ACM, 1981.

[19] Ian Sutherland, "A Model of Information," *Proceedings of the Ninth National Computer Security Conference,* Gaithersburg, MD, 1986.