A Channel-theoretic Account of Separation Security

Gerard Allwein[1] **and William L. Harrison**[2]
[1]Code 5540, Naval Research Laboratory, Washington, D.C. 20375, USA
[2]Department of Computer Science, University of Missouri, Columbia, MO 65211, USA

Abstract—*It has long been held that information flow security models should be organized with respect to a theory of information, but typically they are not. The appeal of a information-theoretic foundation for information flow security seems natural, compelling and, indeed, almost tautological. This article illustrates how channel theory—a theory of information based in logic—can provide a basis for noninterference style security models. The evidence presented here suggests that channel theory is a useful organizing principle for information flow security.*

1. Introduction

It has long been believed that information flow security should be characterized in terms of a theory of Shannon-style [20] information flow. The problem with this is that the quantity "mutual information" is measured; there is no direction. *Channel theory* [3] is a logical or qualitative theory of information flow; direction is represented explicitly and it is capable of supporting a Shannon-theoretic analysis [2] (although we do not do so here). We submit that channel theory is a natural setting for characterizing information flow security policies and mechanisms, and we support it through the specification of a classic model of information flow security within channel theory in an elegant manner.

Most information flow security models are based either on the *noninterference* model of Goguen and Meseguer [7] or on variants of it [21]. Such security models specify end-to-end system security policies in terms of the "views" of the system as a whole by groups of users/processes. Views are typically characterized by partitioning global system inputs and outputs and associating groups of users/processes with these partitions. These input and output partitions determine the view of its associated group. Noninterference-based security policies will require that, for example, changes in a high level security input partition will result in no change to a low level output partition.

Channel theory is also known as the "logic of distributed systems", where "distributed systems" is interpreted in a very broad sense. In channel theory, each subsystem is formulated as a *local logic* which may be intuitively understood as characterizing the subsystem's view of the distributed system as a whole. Connecting these subsystems is a *channel* that governs how theorems in one local logic may be transferred to another local logic. The intuitive parallel with noninterference-based security is explicated here and made formal. The view according to security level is determined here by a local logic and a channel that governs information flow between levels.

By combining several formalisms from logic and semantics, this paper admittedly places demands on the reader beyond what is perhaps usually expected. Channel theory, in particular, is not broadly known and consequently the presentation here must introduce a number of its key concepts before proceeding. The security property we consider—separation—is simple, but as this is the first application of Channel theory to information security, simplicity is a virtue. But more importantly, this paper elaborates the necessary channel theoretic underpinnings (especially co-channels) and sets the stage for further applications of Channel Theory to information security.

Channel theory combines syntax and semantics and hence, the proofs are necessarily semantic in content. This has the sense of soundness from logic, i.e., one shows a particular (Gentzen) sequent holds semantically. The sequents are in a second order logic which quantifies (semantically) over all actions in a program. The combination of the use of co-channels to distribute an invariant and the invariant being a sequent in a simple second order logic is what allows the statement and proof of separation to be so clean.

This article proceeds from research characterizing the classic, noninterference-style security design of Rushby (known as a separation kernel [18]) in terms of monadic language semantics [11], [10]. A *separation kernel*, K (pictured in Figure 1), mediates communication between the high and low level domains, H and L, respectively. Domains H and L may only communicate with the kernel via the channels f and g, respectively. The security policy boils down to demonstrating that inputs to H have no impact on the outputs of L. The security property associated with the separation kernel in Figure 1 is based on the notion that any operation executing in the high security domain H should have no effect whatsoever on the threads executing in the low security domain L.

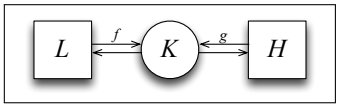


Fig. 1: Sep. Kernel as a Channel Theory Flow Diagram.

In terms of individual atomic operations, this can be further specified as follows. Note that any system execution consists of a sequence of H and L operations determined by some scheduling strategy. The security specification requires that any system execution, $h_0\ ;\ l_0\ ;\ \cdots\ ;\ h_n\ ;\ l_n$, has

precisely the same effect on the L domain as this same execution stripped of H operations, $l_0 \; ; \; \cdots \; ; \; l_n$. This security regime is not, in general, process isolation. Operations in H may not affect those in L, but operations in L are free to influence those in H. Separation requires the converse as well.

The Contributions of This Paper.

Monadic transformers [13], [12] are shown to provide a layering as a scaffold for channel theoretic objects and the relationships among the layers are represented using the morphisms of channel theory (the double-pairs of maps f and g from Figure 1). Through channel theory, by a judicious use of limits, colimits, and a free-variable second-order logic, separation kernels [11], [10] are show to validate the necessary logical statements that express separation. The resulting distributed logical apparatus of channel theory can then be seen to provide an straightforward proof of separation without collapsing the information in the monadic layers into a single but hard to manipulate formal apparatus. The burden of verification is thus made easier.

$$
\begin{array}{ll}
\mathit{StateT}\ \Sigma\ M\ X\ =\ \Sigma \rightarrow M(X \times \Sigma) & \\
\eta_S\ v & = \lambda\sigma.\ \eta_M\,(v, \sigma) \\
x\ \star_S\ f & = \lambda\sigma_0.\,(x\ \sigma_0) \star_M \lambda(v, \sigma_1).\, f\ v\ \sigma_1 \\
\mathsf{g}\ :\ S\ \Sigma & = \lambda\sigma.\ \eta_M\,(\sigma, \sigma) \\
\mathsf{u} : (\Sigma \rightarrow \Sigma) \rightarrow S() & \\
\mathsf{u}\ f & = \lambda\sigma.\ \eta_M((), f\ \sigma) \\
\mathit{lift}\ \varphi & = \lambda\sigma.\ \varphi\ \star_M\ (\eta_M \circ (\lambda v.\,(v, \sigma)))
\end{array}
$$

$$
x \gg y \quad = x\ \star\ \lambda d.\ y \quad \text{— “null” bind}
$$

$$
\begin{array}{ll}
\mathit{ResT}\ M\ X & = \mathrm{fix}\,\xi.\ X\ +\ M\ \xi \\
\eta_R\ v & = \delta\ v \\
(\delta\ x)\ \star_R\ f & = f\ x \\
(\rho\ \varphi)\ \star_R\ f & = \rho\,(\varphi\ \star_M\ \lambda\kappa.\ \eta_M\,(\kappa\ \star_R\ f)) \\
\mathit{step}\ \varphi & = \rho\,(\varphi\ \star_M\ (\eta_M \circ \eta_R)) \\
\mathit{run}\ (\delta\ v) & = \eta_M\ v \\
\mathit{run}\ (\rho\ \varphi) & = \varphi\ \star_M\ \mathit{run}
\end{array}
$$

Fig. 2: Monad Transformers

a) Related Work.: Channel theory is not a logic but, rather, it is a logical framework that allows separation of concerns (in the sense of Dijkstra [5]) at the level of logical specification. It is similar to institutions [6] in that both have similar objects and morphisms. In contrast to institutions, channels are the central organizing principle of channel theory, where channels and co-channels are used for the transfer of theorems between local logics. Separation logic [17] introduces separation of concerns at the level of logical connectives. The channel theoretic characterization of a monadic separation kernel presented here is factored according to the monadic layers underlying the kernel's construction, although channel theory does not focus on monadic specifications exclusively (in contrast to evaluation logic [16], HasCasl [19], or observational program specification [9]). Abadi, et al., [1] formulate notions of dependency (including noninterference) in terms of the *dependency core calculus*. Crary et al. [4] consider a logical characterization of information flow security that, like DCC, has Moggi's computational lambda calculus [14] at its core. The second author's monadic encapsulation of separation [11], [10] is more semantic and model-theoretic than either of these more logical and type-theoretic approaches. The present article answers an open question by making the relationship between such semantic and logical views apparent and precise. Channel theory has not, to the authors' best knowledge, ever been applied to information flow security.

2. Background

This article makes a connection between a number of different formalisms across logic, denotational semantics and information security and, to make it as self-contained as possible, overviews of the necessary background material is presented here. First, noninterference security and separation are discussed and then it is summarized how separation may be realized via monadic language semantics [13], [12]. A brief overview of modular monadic semantics is presented as well. It is assumed of necessity that the reader is familiar with monadic semantics, and, for those wishing more background on monadic semantics, please consult the references. In particular, for an overview of monads for concurrency (i.e., resumption monads), please consult Harrison [8].

Separation Security & Noninterference.

This security regime, which we will refer to as *separation*, is an instance of Goguen and Meseguer noninterference. Separation considers the “static case” where there is no passing of capabilities. Such policies formulate information security in terms of “views” or perspectives with respect to processes or users at the low security level. We assume, without loss of generality that there are precisely two such levels, high and low). If two system inputs, i and i', are the same from the low view (sometimes written $i \approx_L i'$), then, for any system execution consisting of a sequence of high and low operations, $h_0; l_0; \cdots; h_n; l_n$, the following system outputs are the same from the low view: $\mathit{out}\,(h_0; l_0; \cdots; h_n; l_n) \approx_L \mathit{out}\,(l_0; \cdots; l_n)$. Noninterference specifications are typically formulated in terms of abstract state machines.

3. Definition of Channel Theory

The basic structures of channel theory are deceptively simple. The things that are distributed in a distributed system are contexts called *classifications*. The classifications are connected by *infomorphisms*. The relevant definitions follow.

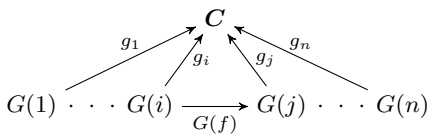
A classification contains two distinct collections of objects, tokens and types. They could be anything that makes sense in using a classification as a model. However, most of modern language theory tends to use the term *types* in a different sense. In this paper, channel theory's types are always termed *propositions*. The tokens are analogous to states or program actions. Bold slanted typeface is always used to denote classifications.

Definition 3.0.1 A *classification*, $\boldsymbol{X}$, is a pair of sets, $Tok(\boldsymbol{X})$, and $Prop(\boldsymbol{X})$, and a relation, $\models_{\boldsymbol{X}} \subseteq Tok(\boldsymbol{X}) \times Prop(\boldsymbol{X})$ written in infix, e.g., $x \models_{\boldsymbol{X}} A$. $x \models_{\boldsymbol{X}} A$ is the qualitative unit of information that flows in channel theory.

Classifications occur wherever models of formal systems are found. Channel theory has its own notion of morphism, called an infomorphism. It is similar to a pair of adjoint functors in that it is a pair of opposing arrows with a condition similar to the adjoint's bijection.

Definition 3.0.2 An *infomorphism* $h : \boldsymbol{X} \rightarrow \boldsymbol{Y}$ of classifications is a pair of contravariant maps, $\overrightarrow{h}$ and $\overleftarrow{h}$ such that $\overrightarrow{h} : Prop(\boldsymbol{X}) \rightarrow Prop(\boldsymbol{Y})$ and $\overleftarrow{h} : Tok(\boldsymbol{Y}) \rightarrow Tok(\boldsymbol{X})$, and for all x and A, the following condition is satisfied, $x^h \models_{\boldsymbol{X}} A$ iff $x \models_{\boldsymbol{Y}} A^h$,. For ease of presentation, $\overleftarrow{h}(x)$ is displayed as x^h and $\overrightarrow{h}(A)$ as A^h.

A commuting *finite cocone* consists of a graph homomorphism G from a finite graph to the category of classifications, a vertex classification $\boldsymbol{C}$, and a collection of arrows $g_i : G(i) \rightarrow \boldsymbol{C}$. It is required that for all $f : i \rightarrow j$, $g_i = g_j \circ G(f)$. The base of the cocone is the objects and arrows identified by G. There will also be a need for cones: a *finite cone* consists of a graph homomorphism G from a finite graph to the category of classifications, a vertex classification $\boldsymbol{C}$, and a collection of arrows $g_i : G(i) \rightarrow \boldsymbol{C}$. It is required that for all $f : j \rightarrow i$, $g_i = G(f) \circ g_j$. Just reverse all the arrows in the preceding diagram.



Definition 3.0.3 An *information channel* is a co-cone in the category of classifications and infomorphisms. An *information co-channel* is a cone in the category of classifications and infomorphisms. $\boldsymbol{C}$ in the diagram is call the core of the (co)channel.

The smallest channel over a base is a colimit. Frequently, the smallest channel is not the most useful because a channel is used as a model. The smallest channel would simply connect the base with no additional modeling apparatus. A colimit in the category of classifications is a colimit on propositions and a limit on tokens.

Assuming a fixed classification $\boldsymbol{C}$, a *Gentzen sequent*, $\Gamma \Vdash_{\boldsymbol{C}} \Delta$ is two sets of propositions connected by a relation $\Vdash$. A *valid* sequent has the force of a meta-level implication of form: for all tokens x, if $x \models_{\boldsymbol{C}} A$ for all of the propositions A in Γ, then at $x \models_{\boldsymbol{C}} B$ for at least proposition in Δ. In this paper, all our sequents will be rather simple and of the form $A \Vdash B$. Sequents are used to represent constraints for a classification. In the core of a channel, sequents underwrite information flow in the core.

Definition 3.0.4 A *local logic* $\mathcal{L} = \langle \boldsymbol{C}, \Vdash_{\mathcal{L}}, N_{\mathcal{L}} \rangle$ consists of a classification $\boldsymbol{C}$, a set $\Vdash_{\mathcal{L}}$ of sequents involving the types of $\boldsymbol{C}$, and a subset $N_{\mathcal{L}} \subseteq Tok(\boldsymbol{C})$ called the *normal tokens* of $\mathcal{L}$, which satisfy all the constraints $\Vdash_{\mathcal{L}}$. A local logic $\mathcal{L}$ is sound if every token is normal; it is complete if every sequent that holds of all normal tokens is in the consequence relation $\Vdash_{\mathcal{L}}$.

Each classification supports a local logic, including cores of channels. A non-normal token represents a counter-example to the theory. In this paper, only normal tokens are used. Non-normal tokens can be used to introduce conditional probabilities associated with the sequents. Typically, the sequents are required to follow certain *structural rules* but these will not concern us in this paper. These non-structural rules allow for the movement of logics forward along an infomorphism $f : \boldsymbol{X} \rightarrow \boldsymbol{Y}$, where Γ^{-f} is an abbreviation for $\overrightarrow{f}^{-1}(\Gamma)$, i.e., the inverse image of Γ under f and Δ^f is the direct image of Δ under f. There are two equivalent forms for each; f-Intro preserves validity and f-Elim preserves non-validity.

$$\frac{\Gamma \Vdash_{\boldsymbol{X}} \Delta}{\Gamma^f \Vdash_{\boldsymbol{Y}} \Delta^f}\,(\mathrm{f}{-}Intro)$$

$$\frac{\Gamma \Vdash_{\boldsymbol{Y}} \Delta}{\Gamma^{-f} \Vdash_{\boldsymbol{X}} \Delta^{-f}}\,(\mathrm{f}{-}Elim)$$

Channels and co-channels are used to hold channel logics. In the core of a channel, a logic can be used to underwrite or authorize information transfer among the side classifications. Co-channels, as they are used in this paper, are used to distribute a common logic in the core to the side classifications.

4. A Channel Account of Separation

The channel theory diagram System Classification (see Figure 3) will be annotated with theorems where $Tok(\boldsymbol{Hi})$ and $Tok(\boldsymbol{Lo})$ are the set of program states for Hi and Lo respectively, $\boldsymbol{Hi}_k$ represents the k-th operation of Hi, $\boldsymbol{C}_H$ is a co-channel which is used for distributing a constraint (expressed as a sequent), $\boldsymbol{H}$ is constructed using the monad $H = StateT\ Hi\ Id$, $\boldsymbol{K}$ is con-

structed using the monad $K = StateT\ Hi\ (StateT\ Lo)\ Id$, $\boldsymbol{R}$ is constructed using the resumption monad. There is a similar diagram below $\boldsymbol{L}$ (using the Lo versions of $\boldsymbol{C}_H$, etc.) as below $\boldsymbol{H}$. Let two program statements be $D := D + 1$ and $D := D - 1$. The first is the Hi operation and the second is the Lo operation. These statements are the actions $D \mapsto D + 1$ and $D \mapsto D - 1$. The other computations needed are $D \mapsto D$ denoted $1_{\boldsymbol{Hi}}$ and $1_{\boldsymbol{Lo}}$ for the respective sides; these are needed for internal housekeeping in the sequel. There are only two $\boldsymbol{Hi}_k$'s in the sample system, namely $\boldsymbol{Hi}_1$ for $1_{\boldsymbol{Hi}}$ and $\boldsymbol{Hi}_2$ for the computational interpretation of $D := D + 1$.

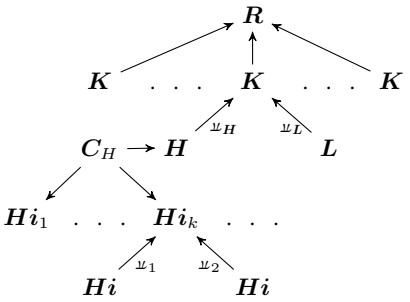


Fig. 3: Sys. Class./Monadic Layers

Notice that the fragment of the diagram containg $\boldsymbol{C}_H, \boldsymbol{Hi}_1, \boldsymbol{Hi}_2, \ldots$ has its arrows pointing downward in contradistinction to the rest of the arrows pointing upward. This is an instance of a cone whereas the other fragments, all of whose arrows have a common target, are cocones. $\boldsymbol{C}_H, \boldsymbol{Hi}_1, \boldsymbol{Hi}_2, \ldots$ is an example of a co-channel. The lone sequent that will be in $\boldsymbol{C}_H$ will be distributed to the $\boldsymbol{Hi}_k$.

Let $\mathfrak{a}_2 = \lambda v.v + 1$, the sequent for $\boldsymbol{Hi}_2$, $\mathcal{U}_1(D = V) \Vdash_{\boldsymbol{Hi}_2} \mathcal{U}_2(D = \mathfrak{a}_2 V)$ will be denoted as: $D = V \Vdash_{\boldsymbol{Hi}_2} \mathfrak{a}_2(D = V)$, eliding the $\mathcal{U}_i$ and making the $\mathfrak{a}$ appear as modal operator. Now for the cone with vertex $\boldsymbol{C}_H$ and base the $\boldsymbol{Hi}_k$. There is an infomorphism (Figure 4) where ϕ_k is the identity map and $Prop(\boldsymbol{C}_H) = Prop(\boldsymbol{Hi}_k)$ for all k. The state pairs $\langle s, s' \rangle$ in $Tok(\boldsymbol{Hi}_k)$ are just those state pairs that satisfy $s \xrightarrow{\mathfrak{a}_k} s'$, where $\xrightarrow{\mathfrak{a}_k}$ symbol means s goes to s' under the action $\mathfrak{a}_k$. Let $\psi_k(\langle s, s' \rangle) = \langle \mathfrak{a}_k.\langle s, s' \rangle \rangle \in Tok(\boldsymbol{C}_H)$. We define $\langle \mathfrak{a}_k, \langle s, s' \rangle \rangle \models_{\boldsymbol{C}_H} Q$ iff $\langle s, s' \rangle \models_{\boldsymbol{Hi}_k} Q$ for any proposition Q. This is a "safe" definition since it will lead to no new sequents holding as constraints in $\boldsymbol{C}_H$ that do not already hold in each of the $\boldsymbol{Hi}_k$; the proposition sets are all identical. This defines $\langle \varphi_k, \psi_k \rangle$ as an infomorphism.

$$\begin{array}{ccc} Prop(\boldsymbol{C}_H) & \xrightarrow{\phi_k} & Prop(\boldsymbol{Hi}_k) \\ \models \Big| & & \Big| \models \\ Tok(\boldsymbol{C}_H) & \xleftarrow[\psi_k]{} & Tok(\boldsymbol{Hi}_k) \end{array}$$

Fig. 4: $\boldsymbol{C}_H$ to $\boldsymbol{Hi}_k$ Infomorphism

To pull the sequent back along ϕ might, given the rules for sequents, result in an invalid sequent in $\boldsymbol{C}_H$. However, notice that this sequent holds for all k under the condition that each $\boldsymbol{Hi}_k$ interprets $\mathfrak{a}$ to the operation $\mathfrak{a}_k$. In this way, the sequent, $D = V \Vdash_{\boldsymbol{C}_H} \mathfrak{a}(D = V)$, becomes a sequent of second-order free-variable logic. However, the sequent still retains its modal character, just as in $\boldsymbol{Hi}_k$. The job of ϕ_k is to distribute this sequent to each $\boldsymbol{Hi}_k$.

Consider each channel $(\boldsymbol{Hi}_k, \boldsymbol{Hi})$ to be a model for a modal logic with the operator $\mathfrak{a}_k$ defining the modality. Each $\boldsymbol{Hi}_K$ contains all pairs of states $\langle s, s' \rangle$ relating s to s' under the action $\mathfrak{a}_k$. Each s and s' must provide a value for D and V. Some values are such $D \neq V$ in which case the antecedent of the sequent is false and hence the sequent evaluates as true; the sequent is then satisfied *spuriously*. In the cases where $D = V$, the consequent follows because $\boldsymbol{Hi}_k$ is the core of the channel for $\mathfrak{a}_k$. Hence the sequent is valid. So $\boldsymbol{Hi}_k$ contains all the models for the logic and the action.

The arrow ψ_k is constructing part of a (flattened) second order model structure in $Tok(\boldsymbol{C}_H)$. It is easier to think of there being a single token in $Tok(\boldsymbol{C}_H)$ for each $\boldsymbol{Hi}_k$. This token needs to pull out a pair from its modal relation to evaluate a proposition, i.e.,

$$\begin{array}{lll} \langle a_k, \langle s, s' \rangle \rangle \models_{\boldsymbol{C}_H} \mathcal{U}_1(D = V) & & \\ \quad \text{iff } \psi_k(\langle s, s' \rangle) \models_{\boldsymbol{C}_H} \mathcal{U}_1(D = V) & & \text{def. of } \psi_k \\ \quad \text{iff } \langle s, s' \rangle \models_{\boldsymbol{Hi}_k} \phi_k(\mathcal{U}_1(D = V)) & & \text{infomorphism} \\ \quad \text{iff } \langle s, s' \rangle \models_{\boldsymbol{Hi}_k} \mathcal{U}_1(D = V) & & \text{def. of } \phi_k \\ \quad \text{iff } \pi_1(\langle s, s' \rangle) \models_{\boldsymbol{Hi}} D = V & & \text{infomorphism} \\ \quad \text{iff } s \models_{\boldsymbol{Hi}} D = V & & \text{def. of } \pi_1 \end{array}$$

In this way, the work load of evaluating propositions, and hence sequents, is distributed via channel theory.

In the sequel, $\mathrm{op}(class)$ will yield the set of operations of the classification $class$. Hence $\mathrm{op}(\boldsymbol{C}_H) = \{\mathfrak{a}_k \mid \langle \mathfrak{a}_k, \langle s, s' \rangle \rangle \in Tok(\boldsymbol{C}_H)\}$. Similarly, $\mathrm{op}(token)$ will yield the operation hiding in a single token.

The token set $Tok(\boldsymbol{C}_H)$ is then a disjoint union of interpretations, each partition contains an entire second-order logic model. The second-order logic used is tightly constrained for the application in this paper. There is no explicit quantifying over function variables (or predicates), hence the moniker *free-variable*. The implicit universal quantification over free function variables is bounded by the classifications needed to evaluate the propositions, it is bounded by the structure of the system being considered.

4.1 The Classifications H and L

We carry out the definitions for the H side, the L side is analogous. The $\boldsymbol{H}$ has the same proposition set as $\boldsymbol{C}_H$ and as tokens pairs of the form $\langle \mathsf{u}_H\ \mathfrak{a}_k, \langle s, \langle (), s' \rangle \rangle \rangle$. The computation $\mathsf{u}_H \mathfrak{a}_k$ is a computation in S. The definitions

$$\mathsf{u}_H\ \langle \mathfrak{a}_k, \langle s, s' \rangle \rangle \stackrel{def}{=} \langle \mathsf{u}_H\ \mathfrak{a}_k, \langle s, s' \rangle \rangle, \quad deHf \stackrel{def}{=} \lambda v.\pi_2(f\ v)$$

(where π_2 projects the second element of a pair) are used to construct the infomorphism from $\boldsymbol{C}_H$ to $\boldsymbol{H}$ which is

the identity on the propositions and is a projection $\pi_{\boldsymbol{C}_H} : Tok(\boldsymbol{H}) \to Tok(\boldsymbol{C}_H)$ on the tokens with the definition

$$\pi_{\boldsymbol{C}_H}\langle \mathfrak{u}_H\ \mathfrak{a}_k, \langle s, \langle (), s'\rangle\rangle \stackrel{def}{=} \langle deH\ (\mathfrak{u}_H\ \mathfrak{a}_k), \langle s, s'\rangle\rangle,$$

where $s \xrightarrow{\mathfrak{a}_k} s'$. It is easily seen that $\pi_{\boldsymbol{C}_H} \circ \mathfrak{u}_H = 1_{Tok(\boldsymbol{C}_H)}$ and $\mathfrak{u}_H \circ \pi_{\boldsymbol{C}_H} = 1_{Tok(\boldsymbol{H})}$. That this is an infomorphism is by stipulation. The proposition sets are the same and the token sets are isomorphic. No new constraints will hence be generated and the old one is preserved (see the $H = StateT\ Hi\ Id$ rule below).

In the sequel, $\mathrm{op}(\boldsymbol{H}) = \{f \mid \langle f, \langle s, \langle (), s'\rangle\rangle \in Tok(\boldsymbol{H})\} \subseteq S()$. Computations used in $\mathrm{op}(\boldsymbol{H})$ and $\mathrm{op}(\boldsymbol{L})$ are of type $a \to t \times a$. The two statements $D := D + 1$ and $D := D - 1$ are compiled into

$$inc = [\![D + 1]\!] \star_s \lambda v.\mathfrak{u}_H(D \mapsto v),$$
$$dec = [\![D - 1]\!] \star_s \lambda v.\mathfrak{u}_L(D \mapsto v)$$

and hence $\mathrm{op}(\boldsymbol{H}) = \{inc, \eta_{\mathfrak{s}}\}$ and $\mathrm{op}(\boldsymbol{L}) = \{dec, \eta_{\mathfrak{s}}\}$. One might consider treating the construction of $\boldsymbol{H}$ to be analogous to the monad transformer $StateT$. The $\boldsymbol{Hi}_k$ is a construction diagram where the projections deconstruct the channel. Thought of in this way, then

$$\frac{D = V \Vdash_{\boldsymbol{C}_H} \mathfrak{a}(D = V)}{D = V \Vdash_{\boldsymbol{H}} \mathfrak{a}(D = V)}\ StateT\ Hi\ Id$$

becomes simply the sequent in $\boldsymbol{C}_H$ moved forward (the sound direction) along an infomorphism.

4.2 The Classification $\boldsymbol{K}$

Let $\boldsymbol{X} \in \{\boldsymbol{H}, \boldsymbol{L}\}$, and for all tokens β in $\boldsymbol{X}$, let $\beta \models_{\boldsymbol{X}} \Phi_{\boldsymbol{X}}$. The sequents

$$\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))$$

will be evaluated using the monad K where $K = StateT\ Hi\ (StateT\ Lo)\ Id$. The tags $\boldsymbol{H}$ and $\boldsymbol{L}$ indicate from which classification the propositions came and are the result of the infomorphisms from $\boldsymbol{H}$ and $\boldsymbol{L}$ being injections on propositions. We let the $\Phi_{\boldsymbol{X}}$ be unaltered by the injections from $\boldsymbol{H}$ and $\boldsymbol{L}$ and note that

$$\Phi_{\boldsymbol{X}} \wedge D = V \Vdash_{\boldsymbol{X}} \mathfrak{x}(D = V)$$

holds in $\boldsymbol{X}$ where $\mathfrak{x} = \mathfrak{a}$ if $\boldsymbol{X} = \boldsymbol{H}$, and $\mathfrak{x} = \mathfrak{b}$ if $\boldsymbol{X} = \boldsymbol{L}$. The operations in K will be restricted to the operations injected by $outer \circ \mathfrak{u}_H$ and $inner \circ \mathfrak{u}_L$.

The construction for the channel $\boldsymbol{K}$ acts like the monad K. Specifically, the sequents of $\boldsymbol{H}$ are operated on by "wrapping them" with $\boldsymbol{L}$ sequents. In our simple case, the sequent transformation can be constructed with the rule:

$$\frac{\begin{array}{l}\Phi_{\boldsymbol{H}} \wedge D = V \Vdash_{\boldsymbol{H}} \mathfrak{a}(D = V) \text{ or} \\ \Phi_{\boldsymbol{L}} \wedge D = U \Vdash_{\boldsymbol{L}} \mathfrak{b}(D = U)\end{array}}{\begin{array}{l}\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \\ \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))\end{array}}\ K$$

This is two rules with Φ_X being appropriately $\Phi_{\boldsymbol{H}}$ or $\Phi_{\boldsymbol{L}}$. The computations necessary for separation are those injected into the monad K using $outer(\mathfrak{u}_H\ \mathfrak{a})$ and $inner(\mathfrak{u}_L\ \mathfrak{b})$. These computations are of type $a \to b \to t \times a \times b$. Tokens in $\boldsymbol{K}$ are of the form $\langle f, \langle s, q, \langle\langle (), s'\rangle, q'\rangle\rangle\rangle$. In our sample system, $f \in \{outer\ inc, outer\ \eta_{\mathfrak{s}}, inner\ dec, inner\ \eta_{\mathfrak{s}}\}$. More generally,

$$\mathrm{op}(\boldsymbol{K}) = \{outer\ f \mid f \in \mathrm{op}(\boldsymbol{H})\} \cup \{inner\ f \mid f \in \mathrm{op}(\boldsymbol{L})\}.$$

The infomorphism $\mathfrak{u}_{\boldsymbol{H}} : \boldsymbol{H} \to \boldsymbol{K}$ has a token morphism $\pi_{\boldsymbol{H}}$.

Definition 4.2.1 Let $prj_H(t, a, b) = (t, a)$ and $prj(t, a, b) = (t, b)$ and

$$deK_H f \stackrel{def}{=} \lambda v.prj_H(f\ v\ undefined),$$
$$deK_L f \stackrel{def}{=} \lambda v.prj_L(f\ v\ undefined),$$

then $\pi_{\boldsymbol{H}} : Tok(\boldsymbol{K}) \to Tok(\boldsymbol{H})$ and $\pi_{\boldsymbol{L}} : Tok(\boldsymbol{K}) \to Tok(\boldsymbol{L})$ are defined as

$$\pi_{\boldsymbol{H}}\langle f, \langle s, q, \langle\langle (), s'\rangle, q'\rangle\rangle \stackrel{def}{=} \langle deK_H\ f, \langle s, \langle (), s'\rangle\rangle\rangle,$$
$$\pi_{\boldsymbol{L}}\langle f, \langle s, q, \langle\langle (), s'\rangle, q'\rangle\rangle \stackrel{def}{=} \langle deK_L\ f, \langle q, \langle (), q'\rangle\rangle\rangle.$$

These definitions reveal the objects $K()$ and $Tok(\boldsymbol{K})$ act like products. Since $\mathrm{op}(\boldsymbol{K}) \subseteq K$, these projections may be restricted to $\mathrm{op}(\boldsymbol{K})$. The classification $\boldsymbol{K}'$ where $Prop(\boldsymbol{K}') = Prop(\boldsymbol{K})$, $Tok(\boldsymbol{K}') = Tok(\boldsymbol{H}) \times Tok(\boldsymbol{L})$ is the co-limit of $\boldsymbol{H}$ and $\boldsymbol{L}$ (recall colimits in the category of classifications entails being a limit on tokens). Hence there is a pair of infomorphisms $k : \boldsymbol{H} \to \boldsymbol{K}'$ and $k' : \boldsymbol{L} \to \boldsymbol{K}'$. It is an easy observation that the restriction of an infomorphism from restricting the domain of the token map is still an infomorphism. Since $Tok(\boldsymbol{K}) \simeq Tok(\boldsymbol{H}) + Tok(\boldsymbol{L})$, the restriction of k and k' to $Tok(\boldsymbol{K})$ yields infomorphisms connecting $\boldsymbol{H}$ with $\boldsymbol{K}$ and $\boldsymbol{L}$ with $\boldsymbol{K}$.

Theorem 4.2.2 *For all $\beta \in Tok(\boldsymbol{K})$, β satisfies*

$$\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))$$

From the fact that the morphisms from $\boldsymbol{H}$ and $\boldsymbol{L}$ to $\boldsymbol{K}$ are infomorphisms, the two sequents

$$\Phi_{\boldsymbol{H}} \wedge \boldsymbol{H}(D = V) \Vdash_{\boldsymbol{K}} \boldsymbol{H}(\mathfrak{a}(D = V)),$$
$$\Phi_{\boldsymbol{L}} \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \boldsymbol{L}(\mathfrak{b}(D = U))$$

are constraints holding in $\boldsymbol{K}$. The usual rules of classical logic then underwrite the theorem.

Theorem 4.2.3

$$deK_H \circ outer = 1_{\mathrm{op}(\boldsymbol{H})};$$
$$deK_L \circ inner = 1_{\mathrm{op}(\boldsymbol{L})}.$$

This theorem says that $outer$ and $inner$ are reversible using the projections from $\mathrm{op}(\boldsymbol{K})$. A consequence of this is that $\mathrm{op}(\boldsymbol{K})$ act like a disjoint sum with $outer$ and $inner$ being the injections:

Corollary 4.2.4 *Let $g : \mathrm{op}(\boldsymbol{H}) \to D$ and $g' : \mathrm{op}(\boldsymbol{L}) \to D$, then there is a unique $k : \mathrm{op}(\boldsymbol{K}) \to D$ such that g factors into $k \circ outer$ and g' into $k \circ inner$.*

Theorem 4.2.5

$$(deK_H \circ \ inner \circ \mathfrak{u}_L)f = \eta_{\mathfrak{s}}();$$
$$(deK_L \circ \ outer \circ \mathfrak{u}_H)f = \eta_{\mathfrak{s}}().$$

This says that injecting a computation from $\mathrm{op}(\boldsymbol{L})$ into $\mathrm{op}(\boldsymbol{K})$ results in a computation whose $\boldsymbol{H}$ component is the identity computation in $\mathrm{op}(\boldsymbol{H})$, i.e., $\eta_{\mathfrak{s}}() = \mathfrak{u}_H \ 1_{Tok(\boldsymbol{H}i_1)}$. A similar statement holds for $\boldsymbol{H}$. Notice that $1_{\mathrm{op}(\boldsymbol{H})} : \mathrm{op}(\boldsymbol{H}) \to \mathrm{op}(\boldsymbol{H})$ while $\eta_{\mathfrak{s}}() \in \mathrm{op}(\boldsymbol{H})$.

Theorem 4.2.6

$$\begin{array}{l}\beta \in \boldsymbol{K} \text{ implies} \\ \quad (\mathrm{op} \circ \pi_{\boldsymbol{H}}(\beta) \neq \eta_S() \text{ and } \mathrm{op} \circ \pi_{\boldsymbol{L}}(\beta) = \eta_S()) \text{ or} \\ \quad\quad (\mathrm{op} \circ \pi_{\boldsymbol{H}}(\beta) = \eta_S() \text{ and } \mathrm{op} \circ \pi_{\boldsymbol{L}}(\beta) \neq \eta_S())\end{array}$$

This theorem is an easy consequence of $outer$ and $inner$ acting like injections to the parameterized disjoint sum $\mathrm{op}(\boldsymbol{K})$ and the projections $\pi_{\boldsymbol{H}}$ and $\pi_{\boldsymbol{L}}$ revealing the components of the elements injected.

Corollary 4.2.7 *For all tokens $\beta \in \boldsymbol{K}$, β satisfying*

$$\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))$$

implies either $\boldsymbol{X} = \boldsymbol{H}$ and $\mathfrak{b} = 1_{\boldsymbol{L}o}$, or $\boldsymbol{X} = \boldsymbol{L}$ and $\mathfrak{a} = 1_{\boldsymbol{H}i}$.

Incidentally, the sequents in the conclusion of the theorem can be seen in model theoretic form in [18] although they were developed independently from the structure of the classifications involved.

4.3 The Classification $\boldsymbol{R}$

The computations $R()$ are on a bijective correspondence with lists of operations from $K()$. The injection $step : K \to R$ with the definition:

$$step \ \ x \stackrel{def}{=} \rho(x \star_K (\eta_{\mathfrak{k}} \circ \delta))$$

is not fundamentally changing the computation x but merely adding some bookkeeping. The following theorem shows that $step$ is reversible:

Theorem 4.3.1 $run \circ step = 1_{S()}$.

Let the infomorphism from $\boldsymbol{K}$ to $\boldsymbol{R}$ be the identity on types. Tokens of $\boldsymbol{R}$ are of the finite and infinite sequences $\langle \sigma_0, \ldots \sigma_{n-1} \rangle, 0 \leq n \leq \infty$ such that each σ_i has the form $\langle step \ \ f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle$ for $\langle f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle$ a token $\boldsymbol{K}$. Also, it is required for

$$\sigma_i = \langle f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle, \quad \text{and}$$
$$\sigma_{i+1} = \langle f', \langle \hat{s}, \hat{q}, \langle\langle (), \hat{s}' \rangle, \hat{q}' \rangle\rangle\rangle,$$

that

(i) $s' = \hat{s}$ and $q' = \hat{q}$ and
(ii) $s(D) = s(V)$ and $q(D) = q(U)$.

(i) allows for only valid computation sequences to appear in $Tok(\boldsymbol{R})$. (ii) allows us to disregard states for which any other values will cause the sequents

$$\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{K}} \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))$$

to hold spuriously by making the antecedent false.

The function $step$ can be extended to work on $Tok(\boldsymbol{K})$ by

$$step\langle f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle \stackrel{def}{=} \langle step \ \ f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle$$

For any tokens $\gamma \in Tok(\boldsymbol{R})$ such that $step \ \ \beta = \gamma$, we let $\gamma \models_{\boldsymbol{R}} Q$ iff $\beta \models_{\boldsymbol{K}} Q$. This will not cause any new propositions to hold in $\boldsymbol{R}$. Now let $\sigma \models_{\boldsymbol{R}} Q$ iff for all $i, \sigma_i \models_{\boldsymbol{R}} Q$. run can be extended and the function run_i defined thusly:

$$run\langle step \ \ f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle \stackrel{def}{=} \langle f, \langle s, q, \langle\langle (), s' \rangle, q' \rangle\rangle\rangle,$$
$$run_i\langle \sigma_0, \ldots \sigma_{n-1} \rangle = run \ \sigma_i, \quad i < n.$$

Theorem 4.3.2 *$\langle step, run_i \rangle$ is an infomorphism for all i.*

The following Lemma says that for any token $\sigma \in \boldsymbol{R}$ is composed of a sequence of actions from the High or Low side and that each High action is paired with the identity for the Low side and visa versa:

Theorem 4.3.3 *For all tokens $\sigma \in \boldsymbol{R}$, for all i, σ_i satisfies either $\mathrm{op} \circ \pi_{\boldsymbol{H}} \circ run(\sigma_i) \neq \eta_{\mathfrak{s}}$, $\mathrm{op} \circ \pi_{\boldsymbol{L}} \circ run(\sigma_i) = \eta_{\mathfrak{s}}$, or $\mathrm{op} \circ \pi_{\boldsymbol{L}} \circ run(\sigma_i) \neq \eta_{\mathfrak{s}}$, $\mathrm{op} \circ \pi_{\boldsymbol{H}} \circ run(\sigma_i) = \eta_{\mathfrak{s}}$.*

The following theorem is valid by pushing the analogous sequent in $\boldsymbol{K}$ forward along the infomorphism from $\boldsymbol{K}$ to $\boldsymbol{R}$ and the previous Lemma.

Corollary 4.3.4 (Separation) *For all tokens $\sigma \in \boldsymbol{R}$, for all i, σ_i satisfies*

$$\Phi_{\boldsymbol{X}} \wedge \boldsymbol{H}(D = V) \wedge \boldsymbol{L}(D = U) \Vdash_{\boldsymbol{R}} \boldsymbol{H}(\mathfrak{a}(D = V)) \wedge \boldsymbol{L}(\mathfrak{b}(D = U))$$

and either $\boldsymbol{X} = \boldsymbol{H}$ and $\mathfrak{b} = 1_{\boldsymbol{L}o}$, or $\boldsymbol{X} = \boldsymbol{L}$ and $\mathfrak{a} = 1_{\boldsymbol{H}i}$.

All that is left to to define that the sequents in the theorem hold of all tokens $\sigma \in \boldsymbol{R}$ just when it holds for all σ_i. These sequents are a second-order invariants over all tokens of $\boldsymbol{R}$; the tokens represent all valid computation sequences augmented with some valuation information for logic formulas. It is by accident that these sequents are satisfied by all the $\sigma \in Tok(\boldsymbol{R})$ as opposed to the individual σ_i. This happened by forcing the condition (ii) in the specification of $Tok(\boldsymbol{R})$. (ii) is necessary to define the infomorphism from $\boldsymbol{K}$ to $\boldsymbol{R}$.

Generalizing from Isolation to Noninterference.

Separation in this paper might more accurately be termed "isolation" where High and Low have no interference with each other. It is easy to change this for High having complete access to its D and low's D and Low having only access to its D. Put quickly, one half of the logical work disappears as there would then be counter-examples to theorems implying High has no access to low's D.

The relation $\approx_L$ of Background can be defined on $Tok(\boldsymbol{R})$ by using the projections taking $Tok(\boldsymbol{R})$ to $Tok(\boldsymbol{K})$ to $Tok(\boldsymbol{L})$ and then projecting out the operation using op. The fact that the high and low variables D are used in this paper is immaterial; they could be replaced by streams of values read from an outside environment. Allowing the k index of $\boldsymbol{Hi}_k$ and $\boldsymbol{Lo}_k$ to be larger provides for more operations than the simple increment and decrement.

5. Conclusion

The channel theory used in the proof of separation was able to track the monadic construction of the system. We feel this is an important organizing principle akin to a natural deduction system. This allows channel theory to distribute the workload of the proof over its classification structure much like a natural deduction system allows one to distribute the workload into subproofs. Each classification was relatively simple. The token sets from $\boldsymbol{H}, \boldsymbol{L}$ and above were extracted from the monad applications. The token sets for $\boldsymbol{Hi}_k, \boldsymbol{Lo}_k$ and below were taken from some pre-monadic and standard Floyd-Hoare soundness conditions. The central driving force for the simple logic statement came from $\boldsymbol{C}_H$ and the recognition that the statement of separation could use a second-order free-variable logic sequent. This sequent arose by using a co-channel to represent the common abstraction leading to the second order sequent.

The choice of second-order free-variable logic simplifies the exposition of separation as a meta-statement about the system. The implicit quantification inherent in any free-variable formal system becomes tamed by the use of channel theory to bound the quantification to operations used in the system.

The system analyzed could have been made more complex without substantial changes to either the overall proof or the way the information flows in the System Classification diagram. There are two flows of interest here, or rather one flow and a lack of flow. Information flows from bottom to top in that diagram. It is a logical flow *about* the system. The lack of flow between $\boldsymbol{H}$ and $\boldsymbol{L}$ is really the meta-statement about the system that is the essence of separation. This was a lack a flow of information *in* the system. Complicating the system with shared kernel variables (or processor registers) would be confined to $\boldsymbol{K}$. Once the basic separation properties could be proven there, they would immediately flow to $\boldsymbol{R}$.

A planned extension is probability analysis. The kernel computations in K might include "leakage". This will be modeled in $\boldsymbol{K}$ using sequents that are not entirely valid, but only partially valid. The required change for channel theory is to include this notion via non-normal tokens in $Tok(\boldsymbol{K})$ which are counter-examples to the theorem expressing non-information flow from $\boldsymbol{H}$ to $\boldsymbol{L}$. Probabilities come either defined mathematically or empirically via tests. If $\boldsymbol{K}$ can be entirely mathematically defined, the probabilities are computed ahead of time. This would allow different design decisions to be made. If the probabilities are empirically determined, say if the system must perform a lot of action with an environment giving rise to the leakage, then steps can be taken for redesign of the system.

Another area of extension is in hardware-software codesign. With a mathematically powerful tool such as monads, the interaction between hardware and software can be abstracted. Channel theory will provide the logical layer for formally proving security properties about such designs. The goal is modular designs where proofs of system properties are similarly modularized by following the monadic structure of the system.

References

[1] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke. A core calculus of dependency. In *Proceedings of the Twenty-sixth ACM Symposium on Principles of Programming Languages*, pages 147–160, January 1999.

[2] G. Allwein A Qualitative Framework for Shannon Information Theories. In *Proceedings of the New Security Paradigms Workshop, 2004*, ACM Press, 2005.

[3] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997. Cambridge Tracts in Theor. Comp. Sci. 44.

[4] K. Crary, A. Kliger, and F. Pfenning. A monadic analysis of information flow security with mutable state. *Journal of Functional Programming*, 15(2), Mar. 2005.

[5] E. W. Dijkstra. My recollections of operating system design. *SIGOPS Oper. Syst. Rev.*, 39(2):4–40, 2005.

[6] J. A. Goguen. Institutions: Abstract model theory for specification and programming. *CLSI Research Reports*, 85-30:1–73, 1985.

[7] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 Symposium on Security and Privacy (SSP '82)*, pages 11–20. IEEE Computer Society Press, Apr. 1990.

[8] W. Harrison. The essence of multitasking. In *11th International Conference on Algebraic Methodology and Software Technology (AMAST 2006)*, pages 158–172, July 2006.

[9] W. Harrison. Proof abstraction for imperative languages. In *Proceedings of the 4th Asian Symposium on Programming Languages and Systems (APLAS06)*, pages 97–113, 2006.

[10] W. Harrison and J. Hook. Achieving information flow security through precise control of effects. In *18th IEEE Computer Security Foundations Workshop (CSFW05)*, pages 16–30, Aix-en-Provence, France, June 2005.

[11] W. Harrison and J. Hook. Achieving information flow security through monadic control of effects. Invited submission to: *Journal of Computer Security*, 2008. 46 pages. Accepted for Publication.

[12] S. Liang. *Modular Monadic Semantics and Compilation*. PhD thesis, Yale University, 1998.

[13] E. Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Dept. of Computer Science, Edinburgh Univ., 1990.

[14] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[15] S. Peyton Jones, editor. *Haskell 98 Language and Libraries, Revised Report*. Cambridge Univ. Press, Apr. 2003.

[16] A. M. Pitts. Evaluation logic. In *Proc. of the IVth Higher Order Workshop*, pages 162–189, 1990.

[17] J. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS'02)*, 2002.

[18] J. Rushby. Proof of separability: A verification technique for a class of security kernels. In *Proceedings of the* 5^{th} *International Symposium on Programming*, pages 352–362, Berlin, 1982. Springer-Verlag.

[19] L. Schäder and T. Mossakowski. *Monad-independent Hoare Logic in HasCasl*, volume Fundamental Approaches to Software Engineering, LNCS 2621, pages 261–277. Springer–Verlag, 2003.

[20] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[21] S. Zdancewic. Challenges for information-flow security. In *Proceedings of the First International Workshop on Programming Language Interference and Dependence (PLID'04)*, 2004.