First-Order Logic for Flow-Limited Authorization

Andrew K. Hirsch
Cornell University
akhirsch@cs.cornell.edu

Pedro H. Azevedo de Amorim
Cornell University
pamorim@cs.cornell.edu

Ethan Cecchetti
Cornell University
ethan@cs.cornell.edu

Ross Tate
Cornell University
ross@cs.cornell.edu

Owen Arden
University of California, Santa Cruz
owen@soe.ucsc.edu

Abstract

We present the Flow-Limited Authorization First-Order Logic (FLAFOL), a logic for reasoning about authorization decisions in the presence of information-flow policies. We formalize the FLAFOL proof system, characterize its proof-theoretic properties, and develop its security guarantees. In particular, FLAFOL is the first logic to provide a non-interference guarantee while supporting all connectives of first-order logic. Furthermore, this guarantee is the first to combine the notions of non-interference from both authorization logic and information-flow systems. All of the theorems in this paper have been proven in Coq.

1 Introduction

Distributed systems often make authorization decisions based on private data, which a public decision might leak. Preventing such leakage requires nontrivial reasoning about the interaction between information flow and authorization policies [4, 5, 8]. In particular, the justification for an authorization decision can violate information-flow policies. To understand this concern, consider a social network where Bob can say that only his friends may view his photos, and that furthermore only his friends may know the contents of his friend list. If Alice is not on Bob's friend list and she tries to view one of his photos, telling her that she does not have permission leaks Bob's private information, specifically that Alice is not on Bob's friend list.

Reasoning about the interaction between information flow and authorization policies is challenging for several reasons. First, authorization logics and information-flow systems use different notions of trust. Information-flow systems tend to focus on tracking data dependencies by representing information-security policies as *labels* on data. They then represent trust as a *flows-to* relation between labels, which determines when one piece of data may safely influence another. In contrast, authorization logics tend to directly encode *delegations* between principals as a *speaks-for* relation. Such delegations are often all-or-nothing, where a delegating principal trusts any statements made by the trusted principal, although some logics (e.g., [10,21,37]) support restricting delegations to specific statements. Flows-to relations implicitly encode delegations while speaks-for relations implicitly encode permitted flows. To understand *how*, we must understand how these disparate notions of trust interact.

Both forms of trust serve to selectively constrain the communication that system components rely on to make secure authorization decisions. For example, in the social network example above, suppose Bob's security settings are recorded on server X, and his photos are stored on server Y. When Alice tries to view Bob's photo, server Y communicates with server X to determine if Alice is permitted to do so. Modeling this communication is important because (1) the servers that Y communicates with influence its authorization decisions, and (2) communication can leak private information.

Describing the information security of authorization decisions such as the one above requires modifying typical authorization policies to include information flow. Information-flow systems are excellent at tracking when and what information one principal communicates to another, specifically by transferring data from one label to another. It is less clear when communications occur in authorization logics. A common approach [1, 24, 37] simply models Alice delegating trust to Bob as Alice importing all of Bob's beliefs.

On the other hand, Authorization logics *do* excel at reasoning about beliefs. Authorization logics allow us to write Alice **says** φ, which means that Alice believes formula φ. Because this **says** statement is itself a formula, we can reason about what Bob believes Alice believes by nesting **says** formulae. Information flow, in contrast, has no notion of belief, and so cannot reason about principals' beliefs about each others' beliefs.

In order to express authorization policies, not only does one need the ability to express trust and communication, but also a battery of propositions and logical connectives. Any tool that combines authorization and information flow should be capable of expressing enough logical connectives to reason about real-world policies. First-order logic seems to be a sweet spot of expressive power: it can encode most authorization policies, but it is still simple enough to have clean semantics. For instance, Nexus [37, 38]—a distributed operating system that uses authorization logic directly in its authorization mechanisms—can encode all of its authorization policies using first-order logic.[1]

Finally, evaluating any attempt to combine authorization and information flow policies must examine the resulting security guarantees. Both authorization logics and information-flow systems have a security property called *non-interference*. Information-flow systems view non-interference as standard, while authorization logics often view it as desirable but unobtainable. Although the two formulations look quite different, both make guarantees limiting how one component of a system can influence—i.e., interfere with—another. In authorization logics, this takes the form "Alice's beliefs can only impact the provability of Bob's beliefs if Bob trusts Alice." In information-flow systems—which are mostly defined over programs—changing the value of an input variable x can only change the value of an output variable y when the label of x flows to the label of y.

Both of these notions of non-interference are important. Consider again the example where Bob's friend list is private but Alice attempts to view his photo. Because Bob's friend list is private, changing the list should not affect Alice's beliefs. For instance, Alice should not be affected by Bob adding or removing Cathy. To enforce this, whether or not Cathy is Bob's friend must not affect the set of Bob's beliefs that Alice *may* learn. This requires authorization-logic non-interference, since Bob's beliefs should not affect Alice's beliefs unless they communicate. It also, however, requires information-flow non-interference, since the privacy of Bob's belief is why he is unwilling to communicate.

Gluing together both ideas of non-interference requires understanding the connection between their notions of trust. As we have discussed, this connection is difficult to formulate, making the non-interference combination harder still.

Our goal in this work is to provide a logic that supports reasoning about both information flow and authorization policies by combining their models of trust to obtain the advantages of both. To this end, we present the *Flow-Limited Authorization First-Order Logic* (FLAFOL), which

- provides a notion of trust between principals that can vary depending on information-flow labels,
- clearly denotes points where communication occurs,
- uses **says** formulae to reason about principals' beliefs, including their beliefs about others' beliefs,
- is expressive enough to encode real-world authorization policies, and
- provides a strong security guarantee which combines both authorization-logic and information-flow non-interference.

[1] The Nexus Authorization Logic is actually a monadic second-order logic, but this is used only to encode speaksfor; their examples only use first-order quantification [37].

We additionally aim to clarify the foundations of flow-limited authorization (introduced by Arden et al. [4]). We therefore strive to keep FLAFOL's model of principals, labels, and communication as simple as possible. For example, unlike previous work, we do not require that labels form a lattice.

A final contribution is development of an implementation of FLAFOL in the Coq proof assistant [28] and formal proofs of all theorems in this paper[2]. Together these consists of 18,384 lines of Coq code in 23 files. We hope that this will form the basis of work using FLAFOL to verify new and existing authorization mechanisms. For more details on the Coq code, see Appendix D.

We are, of course, not the first to recognize the important interaction of information-flow policies with authorization, but all prior work in this area is missing at least one important feature. The three projects that have done the most to combine authorization and information flow are FLAM [4], SecPAL$^+$ [8, 10], and AURA [22, 23]. FLAM models trust using information flow, AURA uses DCC [1, 2], a propositional authorization logic, and SecPAL$^+$ places information flow labels on principal-based trust policies, but does not attempt to reason about the combination at all. Neither FLAM nor SecPAL$^+$ can reason about nested beliefs, and both are significantly restricted in what logical forms are allowed. Finally, FLAM's security guarantees are non-standard and difficult to compare to other languages, while AURA relies on DCC's non-interference guarantee which does not apply on any trust relationships outside of those assumed in the static lattice.

The rest of this paper is organized as follows: In Section 2 we discuss three running examples. This also serves as an intuitive introduction to FLAFOL. In Section 3 we show how FLAFOL's parameterization allows it to model real systems. In Section 4 we detail the FLAFOL proof rules. In Section 5 we discuss the proof theory of FLAFOL, proving important meta-level theorems, including consistency and cut elimination. In Section 6 we provide FLAFOL's non-interference theorem. We discuss future work in Section 7 and related work in Section 8, and finally we conclude in Section 9.

2 FLAFOL By Example

We now examine several examples of authorization policies and how FLAFOL expresses them. This will serve as a gentle introduction to the main ideas of FLAFOL, and introduce notation and running examples we use throughout the paper.

We explore three main examples in this section:

1. Viewing pictures on social media
2. Sanitizing data inputs to prevent SQL injection attacks
3. Providing a hospital bill in the presence of reinsurance

Each setting has different requirements, such as defining the meaning of labels in its own way. The ability of FLAFOL to adapt to each demonstrates its expressive power. In a new setting, it is often convenient—even necessary—to define constants, functions, and relations beyond those baked into FLAFOL. These are straightforward to define since FLAFOL's security guarantee holds for any parameterization. We use such symbols freely in our examples to express our intent clearly. Formally, FLAFOL interprets them using standard proof-theoretic techniques, as we see in Section 3.

Notably, FLAFOL does not allow computation on terms, so the meaning of functions and constants are axiomatized via FLAFOL formulae. This allows principals to disagree on how functions behave, which can be useful in modeling situations where each principal has their own view of some piece of data.

[2]The Coq code is available at `https://github.com/FLAFOL/flafol-coq`.

2.1 Viewing Pictures on Social Media

We begin by reconsidering in more detail the example from Section 1 where Alice requests to view Bob's picture on a social-media service. This service allows Bob to set privacy policies, and Bob made his pictures visible only to his friends. When Alice makes her request, the service can check if she is authorized by scanning Bob's friend list. If she is on the list and the photo is available, it shows her the photo. If she is *not* on Bob's friend list, it shows her an HTTP 403: Forbidden page.

Bob may choose who belongs in the role of "friend." Following the lead of other authorization logics, FLAFOL represents Bob believing that Alice is his friend as $\mathsf{Bob}\ \textsf{\textbf{says}}\ \mathsf{IsFriend}(\mathsf{Alice})$. Since **says** statements can encompass any formula, we can express the fact that Bob believes that Alice is *not* his friend as $\mathsf{Bob}\ \textsf{\textbf{says}}\ \neg\mathsf{IsFriend}(\mathsf{Alice})$.

We interpret these statements as Bob's *beliefs*. This reflects the fact that Bob could be wrong, in the sense that he may affirm formulae with provable negations. There is no requirement that Bob believes all true things nor that Bob only believe true things (see Section 4), so holding an incorrect belief does not require Bob to believe False. Note that because False allows us to prove anything, a principal who *does* believe False will affirm every statement.

Now imagine that, as in Section 1, the social-media service allows Bob to set a privacy policy on his friend list as well. As before, Bob can restrict his friend list so that only his friends may learn its contents. If Alice makes her request and she is on Bob's friend list, she may again see the photo. However, if she is not, showing her an HTTP 403 page would leak Bob's private information; Alice would learn that she is not on Bob's friend list, something Bob only shared with his friends. There is nothing the server can do but hang.

In order to discuss this in FLAFOL, we need a way to express that Bob's friend list is private. Since, formally, his friend list is a series of beliefs about who his friends are, we must express the privacy of those beliefs. We view this as giving each belief a *label* describing Bob's policy about who may learn that belief.

Syntactically, we attach this label to the **says** connective. For example, Bob may use the label $\mathsf{Friends}$ to represent the information-security policy "I will share this with only my friends." If he attaches this policy to the beliefs representing his friend list, there is no way to securely prove either $\mathsf{Bob}\ \textsf{\textbf{says}}_\ell\ \mathsf{IsFriend}(\mathsf{Alice})$ or $\mathsf{Bob}\ \textsf{\textbf{says}}_\ell\ \neg\mathsf{IsFriend}(\mathsf{Alice})$ when ℓ is less restrictive than $\mathsf{Friends}$; FLAFOL's security guarantee (Theorem 5) shows that every FLAFOL proof is secure, so neither option is provable in FLAFOL. Because the system searches for one proof to show Alice the picture, and the other to show her the 403 page, hanging is its only option.

A better design of the social-media service might reject policies where the only secure behavior is to hang. Such a system would only allow policies $P(\vec{x})$ where it can either prove $P(\vec{t})$ or prove $\neg P(\vec{t})$ for any list of terms $\vec{t}$. That is, the predicate P must be decidable. Since FLAFOL is intuitionistic, this is the same as a FLAFOL proof that $\forall \vec{x}, P(\vec{x}) \vee \neg P(\vec{x})$.

To avoid hanging in response to Alice's request, the social-media service thus needs the predicate $\mathsf{Bob}\ \textsf{\textbf{says}}_\ell\ \mathsf{IsFriend}(\mathsf{Alice})$ to be decidable at some ℓ that Alice can read. Unfortunately, both options leak information about Bob's friend list—which is restricted to $\mathsf{Friends}$—and all FLAFOL proofs are secure, so it must be undecidable. If Bob's friend list were public, simply checking the list would be enough to decide this predicate. FLAFOL can easily express this by labeling each of Bob's beliefs about $\mathsf{IsFriend}$ as Public.

Another, more subtle, change would be to say that every principal can find out whether *they* are on Bob's friend list, but only Bob's friends can see the rest of the list. FLAFOL can also express this policy and prove it decidable, but doing so would require significant infrastructure using the technology we will build in Sections 3 and 4.

This example also demonstrates both how naively reasoning about authorization with information flow can cause leaks and how FLAFOL can help reason about those beliefs.

2.2 Integrity Tracking to Prevent SQL Injection

For our second example, imagine a stateful web application. It takes requests, updates its database, and returns web pages. In order to avoid SQL injection attacks, the system will only update its database based on high-integrity input. However, it marks all web request inputs as low integrity, representing the fact that they may contain attacks. The server has a sanitization function San that will neutralize attacks, so when it encounters a low-integrity input, it is willing to sanitize that input and endorse the result.

FLAFOL's support for arbitrary implications allows it to easily encode such endorsements. Let the predicate $\mathsf{DBInput}(x)$ mean that a value x—possibly taken from a web request—is a database input. When a user makes a request with database input x, we can thus represent it as $\mathsf{System}\ \textbf{says}_{\mathsf{LInt}}\ \mathsf{DBInput}(x)$. Here LInt represent low-integrity beliefs. We represent the system's willingness to endorse any sanitized input as:

$$\mathsf{System}\ \textbf{says}_{\mathsf{LInt}}\ \mathsf{DBInput}(x) \to \mathsf{System}\ \textbf{says}_{\mathsf{HInt}}\ \mathsf{DBInput}(\mathsf{San}(x))$$

This example shows the power of arbitrary implications for expressing authorization and information-flow policies. It also, however, demonstrates their dangers, since unconstrained downgrades can allow information to flow in unintended ways. In Section 6 we will discuss how our non-interference theorem (Theorem 5) adapts to these downgrades by weakening its guarantees.

2.3 Hospital Bills Calculation and Reinsurance

Imagine now that Alice finds herself in the hospital. Luckily she has employer-provided insurance, but her employer just switched insurance companies. Now she has two unexpired insurance cards, and she cannot figure out which one should be paying for this operation. Thus, either of two insurers, I_1 and I_2, may be paying.

Imagine further that Bob's job is to create a correct hospital bill for Alice. He uses the label ℓ_H to determine both who may learn the contents of Alice's bill and who may help determine them. That is, ℓ_H expresses both a confidentiality policy and an integrity policy. Bob believes that Alice's insurer may help determine the contents of Alice's bill, since they can decide how much they are willing to pay for Alice's surgery.

Bob knows that I_2 has a reinsurance treaty with I_1. This means that if Alice is insured with I_2 and the surgery is very expensive, I_1 will pay some of the bill. Thus, I_1 may help determine the contents of Alice's hospital bill, even if I_2 turns out to be her current insurer.

Bob is willing to accept Alice's insurance cards as evidence that she is insured by either I_1 or I_2, which we can express as $\mathsf{Bob}\ \textbf{says}_{\ell_H}\ (\mathsf{CanWrite}(I_1, \ell_H) \vee \mathsf{CanWrite}(I_2, \ell_H))$. Because Bob knows about I_2's reinsurance treaty with I_1, he knows that if I_2 helps determine the contents of Alice's bill, they will delegate some of their power to I_1, which we express as $\mathsf{Bob}\ \textbf{says}_{\ell_H}\ (I_2\ \textbf{says}_{\ell_H}\ \mathsf{CanWrite}(I_1, \ell_H))$.

Bob's beliefs allow him to prove that I_1 may help determine the contents of Alice's bill, since by assuming the previous two statements we can prove that $\mathsf{Bob}\ \textbf{says}_{\ell_H}\ \mathsf{CanWrite}(I_1, \ell_H)$. There are two possible cases: if Bob already believes that I_1 can help determine the contents of Alice's bill, we are done. Otherwise, Bob believes that I_2 can help determine the contents of Alice's bill, and so Bob is willing to let I_2 delegate their power. Since he knows that they will delegate their power to I_1, he knows that I_1 can help determine the contents of Alice's bill in this case as well. This covers all of the cases, so we can conclude that $\mathsf{Bob}\ \textbf{says}_{\ell_H}\ \mathsf{CanWrite}(I_1, \ell_H)$.

We think of Bob as performing this proof, since it is entirely about Bob's beliefs. From this point of view, Bob's ability to reason about I_2's beliefs appears to be Bob *simulating* I_2. This ability of one principal to simulate another provides the key intuition to understand the *generalized principal*, a fundamental construct in the formal presentation of FLAFOL (see Section 3).

We also note that Bob used I_2's beliefs in this proof, even though he does not necessarily trust I_2. However, he *might* trust it if it turns out to be Alice's insurer. Because Bob trusts I_2 in part of the proof but not in general, we refer to this as *discoverable trust*. FLAFOL's ability to handle discoverable trust makes reasoning about its security properties more difficult, as we see in Section 6.

This example shows how disjunctions can be used to express policies when principals do not know the state of the world. It also demonstrates how disjunctions make it difficult to know how information can flow at any point in time, since we may discover new statements of trust under one branch of a disjunction. FLAFOL's non-interference theorem adapts to this by considering all declarations of trust that could possibly be discovered in a given context.

2.4 Further Adapting FLAFOL

All of the above examples use information-flow labels to express confidentiality policies, integrity policies, or both. While confidentiality and integrity are mainstay features of information flow tracking, information-flow labels can also express other properties. For instance, MixT [29] describes how to use information-flow labels to create safe transactions across databases with different consistency models, and the work of Zheng and Myers [45] uses information-flow labels to provide availability guarantees. FLAFOL allows such alternative interpretations of labels by using an abstract *permission model* to give meaning to labels.

By default, the permissions gain meaning only through their behavior in context, but they are able to encode and reason about a wide variety of authorization mechanisms. In Section 3, we see how FLAFOL can be used to reason about capabilities, and in Appendix A we discuss a model closer to military classification.

3 Using FLAFOL

In this section, we examine how to use FLAFOL to reason about real systems. To do this, we look at a fictional verified distributed-systems designer Dana. She wants to formally prove that confused-deputy attacks are impossible in her capability-based system with copyable, delegatable read capabilities. Dana employs a six-step process to reason about her system in FLAFOL:

1. Decide on a set $\mathcal{S}$ of *sorts* of data she wants to represent.
2. Choose a set $\mathcal{F}$ of *function symbols* representing operations in the system, and give those operations types.
3. Choose a set $\mathcal{R}$ of *relation symbols* representing atomic facts to reason about, and give those relations types.
4. Develop axioms that encode meaning for these relationships.
5. Specify meta-level theorems stating her desired properties.
6. Prove that those meta-level theorems hold.

Sorts. First, Dana decides on what sorts of data she wants to represent. We can think of *sort* as the logic word for "type." FLAFOL is defined with respect to a set $\mathcal{S}$ of sorts that must include at least Label and $\mathsf{Principal}$, but may contain more. Dana wants to reason about capability tokens that grant read access to data, so she also includes a sort named Token.

Dana uses the $\mathsf{Principal}$ sort to represent system principals, but conceptually divides the Label sort into $\mathsf{Confidentiality}$ and $\mathsf{Integrity}$. Each $\mathsf{Confidentiality}$ value defines a confidentiality policy which may be applied to many pieces of data. A capability (which is always public itself) grants read access to data governed by one or more such policies. She uses the $\mathsf{Integrity}$ sort to represent integrity policies on tokens themselves. We will see below how she can enforce $\mathsf{Label} = \mathsf{Confidentiality} \times \mathsf{Integrity}$.

Function Symbols. Dana next decides on operations she wants to reason about. This is also her chance to define constants using nullary operations. Formally, FLAFOL is defined with respect to an arbitrary set $\mathcal{F}$ of *function symbols*. Each function comes equipped with a *signature*, or type, expressing when it can be applied.

Dana thinks about what information she needs to access about a given token. She needs to be able to determine the confidentiality a token grants permission to read, the integrity of that token, and which principal is the token's *root of authority*—that is, who created the token. She thus creates three function symbols, $\mathsf{TknConf} : \mathsf{Token} \to \mathsf{Confidentiality}$, $\mathsf{IntegOfTkn} : \mathsf{Token} \to \mathsf{Integrity}$ and $\mathsf{RootOfAuth} : \mathsf{Token} \to \mathsf{Principal}$. She also needs to be able to determine the integrity that a principal commands, so she includes a function symbol $\mathsf{IntegOf} : \mathsf{Principal} \to \mathsf{Integrity}$. Finally, since a token can potentially be transferred to anyone in her system, she creates a constant $\mathsf{Public} : \mathsf{Confidentiality}$ to represent this.

Dana wants to enforce that labels are pairs of confidentiality and integrity. She therefore creates two "projection" function symbols $\pi_{\mathsf{C}} : \mathsf{Label} \to \mathsf{Confidentiality}$ and $\pi_{\mathsf{I}} : \mathsf{Label} \to \mathsf{Integrity}$, along with a third pair symbol $(\_, \_) : \mathsf{Confidentiality} \to \mathsf{Integrity} \to \mathsf{Label}$. The first two ensure that labels contain a confidentiality and an integrity, while pairing allows creation of labels from a confidentiality with an integrity. This makes labels pairs of confidentiality and integrity.[3]

Relation Symbols. Dana can now choose relations representing facts that she wants to reason about. Along with sorts and functions, FLAFOL is defined with respect to a set $\mathcal{R}$ of *relation symbols*, allowing it to reason about more facts. The set $\mathcal{R}$ must include at least flows-to ($\sqsubseteq$), $\mathsf{CanRead}$, and $\mathsf{CanWrite}$, but may contain more. We call these required relations *permissions* because they define the trust relationships governing communication. The relation $\ell \sqsubseteq \ell'$ means information with label ℓ can affect information with label ℓ', $\mathsf{CanRead}(p, \ell)$ means that principal p may learn beliefs with label ℓ, and $\mathsf{CanWrite}(p, \ell)$ means p may influence beliefs with label ℓ.

Dana is able to use these relations to define the permissions her capability tokens grant. She also includes a fourth relation in $\mathcal{R}$, $\mathsf{HasToken}(\mathsf{Principal}, \mathsf{Token})$, defining token possession: if $\mathsf{HasToken}(p, t)$, then principal p has (a copy of) token t.

Axioms. Dana describes the behavior of her system with axioms that use the sorts, functions, and relations she defined above. These should be *consistent*, in the sense that they do not allow a derivation of False. Theorem 1 in Section 5.1 gives conditions under which all of the axioms that we will discuss in this section are consistent.

Dana uses three main axioms: one describing how tokens may be copied and delegated, one describing when one principal may read another's beliefs, and one describing when a principal may affect another's beliefs. She may use more axioms if she likes—e.g., to capture principals' beliefs about permitted flows between labels.

Dana's first axiom allows any principal to copy any capability it holds and give that copy to another principal:

$$\begin{array}{l} \forall q : \mathsf{Principal}.\, \forall t : \mathsf{Token}. \\ \left(\begin{array}{l} \exists p : \mathsf{Principal}.\, \mathsf{HasToken}(p, t) \wedge \\ p \ \textbf{says}_{(\mathsf{Public}, \mathsf{IntegOfTkn}(t))} \ \mathsf{HasToken}(q, t) \end{array} \right) \to \mathsf{HasToken}(q, t) \end{array}$$

This says that, for principals p and q, if p holds a read capability token t, p can pass t to q. To do so, p must affirm that q has t at a public label with the integrity of the token. The public label requires Dana to ensure that, when one principal copies a token and passes it to another principal, everyone is allowed to know this information.

Dana's second axiom defines when a principal p allows q to read a belief of p's labeled ℓ. First, p checks that q has a token, and that p believes that the token gives read access to something at least as confidential as

[3] Technically, Dana also needs to write axioms equivalent to the η and β laws for pairs in order to labels to truly be pairs. However, this is a technical point that does not add to the current discussion, so we elide this here.

Sorts	σ	$::=$	$\textsf{Label} \mid \textsf{Principal} \mid \cdots$
Labels	ℓ		
Principals	p, q, r		
Functions	f	$::=$	$\cdots$
Relations	R	$::=$	$\textsf{CanRead}(\textsf{Principal}, \textsf{Label})$
		$\mid$	$\textsf{CanWrite}(\textsf{Principal}, \textsf{Label})$
		$\mid$	$\textsf{Label} \sqsubseteq \textsf{Label} \mid \cdots$
σ-terms	t	$::=$	$x \mid f(t_1, \ldots, t_n)$
Formulae	φ, ψ, χ	$::=$	$R(t_1, \ldots, t_n)$
		$\mid$	$\textsf{True} \mid \textsf{False}$
		$\mid$	$\varphi \land \psi \mid \varphi \lor \psi \mid \varphi \to \psi$
		$\mid$	$\forall x\!:\!\sigma.\, \varphi \mid \exists x\!:\!\sigma.\, \varphi$
		$\mid$	$p \textbf{ says}_\ell\, \varphi$
Generalized Principals	g	$::=$	$\langle\rangle \mid g \cdot p\langle \ell \rangle$

Figure 1: FLAFOL Syntax

ℓ. Second, p checks to make sure that the token's root authority may influence this belief:

$$\begin{array}{l}\forall q\!:\!\textsf{Principal}.\, \forall \ell\!:\!\textsf{Label}.\, \forall p\!:\!\textsf{Principal}.\, \forall \ell'\!:\!\textsf{Label}.\\ \quad \left(\begin{array}{l}\exists t\!:\!\textsf{Token}.\, \textsf{HasToken}(q, t)\\ \qquad\qquad \land\, p \textbf{ says}_{\ell'}\, \pi_{\textsf{C}}(\ell) \sqsubseteq \textsf{TknConf}(t)\\ \qquad\qquad \land\, p \textbf{ says}_{\ell'}\, \textsf{CanWrite}(\textsf{RootOfAuth}(t), \ell')\end{array}\right)\\ \to p \textbf{ says}_{\ell'}\, (\textsf{CanRead}(q, \ell))\end{array}$$

More formally, it says that if q holds some token t and p believes both that t grants read permissions for ℓ's confidentiality and that the root of authority for t can influence p's beliefs at ℓ', then p will allow q to read ℓ. This defines what it means for a principal (p here) to believe that a token grants read access to their data. Dana now needs to make sure that whenever a read access is granted in her system, not only does the principal who gets read access have a token, but that the principal who owns the data does indeed believe that the token grants read access to that data.

Finally, her third axiom states that a principal can write a label (according to a second principal) if the integrity of that principal flows to the integrity of the label:

$$\begin{array}{l}\forall q\!:\!\textsf{Principal}.\, \forall \ell\!:\!\textsf{Label}.\, \forall p\!:\!\textsf{Principal}.\, \forall \ell'\!:\!\textsf{Label}.\\ \quad p \textbf{ says}_{\ell'}\, (\textsf{IntegOf}(q) \sqsubseteq \pi_{\textsf{I}}(\ell)) \to p \textbf{ says}_{\ell'}\, (\textsf{CanWrite}(q, \ell))\end{array}$$

Dana then needs to make sure that write accesses are only granted to principals with high enough integrity.

Metatheoretic Properties. Dana has now created a model of her system, so she can use it to state and prove properties of her system as meta-theorems. Luckily, Rajani, Garg, and Rezk [34] have shown that information-flow integrity tracking with a non-interference result is sufficient to avoid confused deputy attacks with capability systems. Therefore Theorem 5 provides the guarantees she needs.

FLAFOL Syntax. This example demonstrates FLAFOL's flexibility as a powerful tool for reasoning about authorization mechanisms in the presence of information-flow policies. We saw that, since FLAFOL is defined with respect to the three sets $\mathcal{S}$, $\mathcal{F}$, and $\mathcal{R}$, it can express the key components of a system. This parameterized definition gives rise to the formal FLAFOL syntax in Figure 1.

$$\textsc{FlowsToRefl}\ \frac{}{\Gamma \vdash \ell \sqsubseteq \ell \ @\ g} \qquad \textsc{FlowsToTrans}\ \frac{\Gamma \vdash \ell_1 \sqsubseteq \ell_2 \ @\ g \qquad \Gamma \vdash \ell_2 \sqsubseteq \ell_3 \ @\ g}{\Gamma \vdash \ell_1 \sqsubseteq \ell_3 \ @\ g}$$

$$\textsc{CRVar}\ \frac{\Gamma \vdash \mathsf{CanRead}(p, \ell_2) \ @\ g \qquad \Gamma \vdash \ell_1 \sqsubseteq \ell_2 \ @\ g}{\Gamma \vdash \mathsf{CanRead}(p, \ell_1) \ @\ g}$$

$$\textsc{CWVar}\ \frac{\Gamma \vdash \mathsf{CanWrite}(p, \ell_1) \ @\ g \qquad \Gamma \vdash \ell_1 \sqsubseteq \ell_2 \ @\ g}{\Gamma \vdash \mathsf{CanWrite}(p, \ell_2) \ @\ g}$$

Figure 2: Permission Rules

In order to use the function and relation symbols and incorporate axioms, FLAFOL allows proofs to occur in a context. FLAFOL additionally includes rules requiring flows-to to be reflexive and transitive, placing a preorder on the Label sort,[4] and requiring CanRead and CanWrite to respect a form of variance. If $\ell_1 \sqsubseteq \ell_2$ and Alice can read data A with label ℓ_2, then she may learn information about data with label ℓ_1 used to calculate A. This means she should also be able to read data with label ℓ_1. Thus, CanRead must (contravariantly) respect the preorder on labels. Similarly, if Alice can help determine some piece of data B labeled with ℓ_1, she can influence any data labeled with ℓ_2 that is calculated from B, so Alice should be able to help determine data labeled at ℓ_2. Thus, CanWrite must (covariantly) respect the preorder on labels.

Figure 2 presents these rules formally. We give the proof rules in the form of a sequent calculus. The trailing $@\ g$ represents *who* affirms that formula in the proof, similarly to how **says** formulae represent who affirms a statement at the object level. Unlike **says** formulae, these meta-level objects—called *generalized principals*—encode arbitrary reasoners, including possibly-simulated principals.

Recall from Section 2.3 that we can think of some proofs as being performed by principals if those proofs entirely involve that principal's beliefs. In that example, Bob reasoned about his belief that another principal, the insurer I_2, trusted a third principal, the insurer I_3. We think of this ability to reason about the beliefs of others as the ability to *simulate* other principals. In fact, because principals' beliefs are segmented by labels, principals can have multiple simulations of the same other principal.

This suggests that FLAFOL captures the reasoning of principals *at some level of simulation*. A generalized principal is a stack of principal/label pairs, representing a stack of simulators and simulations. The empty stack, written $\langle\rangle$, represents *ground truth*. A stack with one more level, written $g \cdot p\langle\ell\rangle$, represents the beliefs of p at level ℓ according to the generalized principal g. Figure 1 contains the formal grammar for generalized principals.

4 Proof System

So far, we have discussed the intuitions behind FLAFOL and its syntax. Here we introduce FLAFOL formally. Unfortunately, we cannot examine every aspect of FLAFOL's formal presentation in detail, though interested readers should see Appendix F. Instead, we discuss the most novel and most security-relevant aspects of FLAFOL's design.

FLAFOL sequents are of the form $\Gamma \vdash \varphi \ @\ g$, where Γ is a context containing beliefs. This means that the FLAFOL proof system manipulates beliefs, as described in Section 3. Readers familiar with sequent

[4] Many information-flow tools require their labels to form a lattice. We find that a preorder is sufficient for FLAFOL's design and guarantees, so we decline to impose additional structure. In Section 5.1 we show that enforcing a lattice structure is both simple and logically consistent.

$$\textsc{FalseL}\ \frac{}{\Gamma, \textsf{False} \mathbin{@} g \vdash \varphi \mathbin{@} g \cdot g'}$$

$$\textsc{OrL}\ \frac{\Gamma, \varphi \mathbin{@} g \vdash \chi \mathbin{@} g' \qquad \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{OrR1}\ \frac{\Gamma \vdash \varphi \mathbin{@} g}{\Gamma \vdash \varphi \vee \psi \mathbin{@} g} \qquad \textsc{OrR2}\ \frac{\Gamma \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \vee \psi \mathbin{@} g}$$

$$\textsc{ImpL}\ \frac{\Gamma \vdash \varphi \mathbin{@} \langle\rangle \qquad \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \to \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{ImpR}\ \frac{\Gamma, \varphi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \to \psi \mathbin{@} g}$$

$$\textsc{SaysL}\ \frac{\Gamma, \varphi \mathbin{@} g \cdot p\langle\ell\rangle \vdash \psi \mathbin{@} g'}{\Gamma, p \ \textbf{says}_\ell\ \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'} \qquad \textsc{SaysR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle}{\Gamma \vdash p \ \textbf{says}_\ell\ \varphi \mathbin{@} g}$$

$$\textsc{VarR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell'\rangle \cdot g' \qquad \Gamma \vdash \ell' \sqsubseteq \ell \mathbin{@} g \cdot p\langle\ell\rangle}{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g'}$$

$$\textsc{FwdR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \qquad \Gamma \vdash \textsf{CanRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \qquad \Gamma \vdash \textsf{CanWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle}{\Gamma \vdash \varphi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g'}$$

Figure 3: Selected FLAFOL Proof Rules

calculus may recognize that FLAFOL is intuitionistic, as there is only one belief on the right side of the turnstile.

Sequent calculus rules tend to manipulate beliefs either on the left or the right side of the turnstile. For instance, consider the FLAFOL rules for conjunctions:

$$\textsc{AndL}\ \frac{\Gamma, (\varphi \mathbin{@} g), (\psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \wedge \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{AndR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \qquad \Gamma \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \wedge \psi \mathbin{@} g}$$

We find it easiest to read left rules "up" and right rules "down." With this reading, the AndL rule uses an assumption of the form $\varphi \wedge \psi \mathbin{@} g$ by splitting it into two assumptions, one for each conjunct, while the AndR rule takes proofs of two formulae and proves their conjunction.[5]

Most of the rules of FLAFOL are standard rules for first-order logic with generalized principals included to indicate who believes each formula. For instance, the rules for conjunctions above were likely familiar to those who know sequent calculus.

Figure 3 contains FLAFOL rules selected for discussion. The first, FalseL, tells us how to use False as an assumption. In standard intuitionistic first-order logic, this is simply the principle of Ex Falso: if we assume False, we can prove anything. In FLAFOL, a generalized principal who assumes false is willing to affirm any formula. This includes statements about other principals, so FalseL extends the generalized principal arbitrarily. We use $g \cdot g'$ as notation for extending the generalized principal g with a list of principal-label pairs, denoted g'.

We discuss the disjunction rules OrR1, OrR2, and OrL because **says** distributes over disjunctions. That is, given $p \ \textbf{says}_\ell\ (\varphi \vee \psi)$, we can prove $(p \ \textbf{says}_\ell\ \varphi) \vee (p \ \textbf{says}_\ell\ \psi)$. In an intuitionistic logic like FLAFOL,[6] disjunctions must be a proof of one side or the other. The proof of distribution of **says** over $\vee$ then says that if p has evidence of either φ or ψ, then p can examine this evidence to discover whether it is evidence of φ or evidence of ψ.

[5]For readers interested in learning more about sequent calculus, we recommend MIT's interactive tool for teaching sequent caluclus as a tutorial [43].

[6]Recall that we argued in Section 2.1 that reasoning about authorization and information-flow security together is naturally intuitionistic.

The implication rules ImpR and ImpL interpret the premise of an implication as ground truth, while the generalized principal who believes the implication believes the consequent. In particular, this means that **says** statements do not distribute over implication as one might expect, i.e., p **says**$_\ell$ $(\varphi \to \psi)$ does not imply that $(p$ **says**$_\ell$ $\varphi) \to (p$ **says**$_\ell$ $\psi)$. Instead, p **says**$_\ell$ $(\varphi \to \psi)$ implies $\varphi \to (p$ **says**$_\ell$ $\psi)$. We can thus think of implications as *conditional* knowledge. That is, if a generalized principal g believes $\varphi \to \psi$, then g believes ψ conditional on φ being true about the system.

We can still form implications about generalized principals' beliefs, but we must insert appropriate **says** statements into the premise to do so. In Section 5.5, we discuss how this semantics is necessary for both our proof theoretic and security results.

The next two rules of Figure 3, SaysR and SaysL, are the only rules which specifically manipulate **says** formulae. Essentially, generalized principals allow us to delete the **says** part of a formula while not forgetting who said it. Thus, generalized principals allow us to define sequent calculus rules once for every possible reasoner.

The final rules, VarR and FwdR, define communication in FLAFOL. Both manipulate beliefs on the right, have corresponding left rules, which acts contravariantly and can be found in Appendix F.

Information-flow communication is provided by the variance rule VarR. This can be thought of like the variance rules used in subtyping. Most systems with information-flow labels do not have explicit variance rules, but instead manipulate relevant labels in every rule. By adding an explicit variance rule, we not only simplify every other FLAFOL rule, we also remove the need for the label join and meet operators that are usually used to perform the label manipulations. Others have noted that adding explicit variance rules improves the design of the rest of the system [3, 41], but it remains an unusual choice.

The forwarding rule Fwd provides authorization-logic-style communication. In FLAFOL, p can forward a belief at label ℓ to q if:

- p is willing to send its beliefs at label ℓ to q, denoted p **says**$_\ell$ $\mathsf{CanRead}(q, \ell)$, and
- q is willing to allow p to determine its beliefs at label ℓ, denoted q **says**$_\ell$ $\mathsf{CanWrite}(p, \ell)$.

After establishing this trust, p can package up its belief and send it to q, who will believe it at the same label.

5 Proof Theory

In this section, we evaluate FLAFOL's logical design. We show that FLAFOL has the standard sequent calculus properties of (positive) consistency and cut elimination and discuss fundamental limitations that inform our unusual implication semantics. We also develop a new proof-theoretic tool, *compatible supercontexts*, for use in our non-interference theorem in Section 6.

5.1 Consistency

One of the most important properties about a logic is consistency, meaning it is impossible to prove False. This is not possible in an arbitrary context, since one could always assume False. One standard solution is to limit the theorem to the empty context. By examining the FLAFOL proof rules, however, we see that it is only possible to prove False by assumption or by Ex Falso. Either method requires that False already be on the left-hand side of the turnstile, so if False can never get there, then it should be impossible to prove.

To understand when False can appear on the left-hand side of the turnstile, we note that formulae on the left tend to stay on the left and formulae on the right tend to stay on the right. The only exception is the implication rules ImpL and ImpR which move the premise of the implication to the other side. The fact that no proof rule allows us to change either side of the sequent arbitrarily gives useful structure to proofs. To handle implications, however, we must keep track of their nesting structure, which we do by considering

$$s \in \{+, -\} \qquad \overline{+} = - \qquad \overline{-} = +$$

$$\frac{}{\varphi^s \leq (\varphi \land \psi)^s} \qquad \frac{}{\psi^s \leq (\varphi \land \psi)^s}$$

$$\frac{}{\varphi^{\overline{s}} \leq (\varphi \to \psi)^s} \qquad \frac{}{\psi^s \leq (\varphi \to \psi)^s}$$

$$\frac{}{\varphi^s \leq \varphi^s} \qquad \frac{\varphi^s \leq \psi^{s'} \qquad \psi^{s'} \leq \chi^{s''}}{\varphi^s \leq \chi^{s''}} \qquad \frac{}{\varphi^s \leq (p \ \textsf{\textbf{says}}_\ell \ \varphi)^s}$$

Figure 4: Selected rules for the Signed Subformula Relation

signed formulae. We call a formula in a sequent *positive* if it appears on the right side of the turnstile and *negative* if it appears on the left. If φ is positive we write φ^+, and if φ is negative we write φ^-. Figure 4 shows selected rules from the *signed subformula relation*, which we discuss in more depth in Appendix B.

The intuition above and this relation lead to the following theorem. Note that formulae which do not contain False as a negative subformula are called *positive* formulae, explaining the name.

Theorem 1 (Positive Consistency)**.** *For any context* Γ*, if*

$$\textsf{False}^- \not\leq \varphi^- \textit{ for all } \varphi \ @ \ g \in \Gamma$$

then $\Gamma \nvdash \textsf{False} \ @ \ g'$.

The proof follows by induction on the FLAFOL proof rules.

We get the result with an empty context as a corollary. This states that False is not a theorem of FLAFOL.

Corollary 1 (Consistency)**.** $\nvdash \textsf{False} \ @ \ g$

Proof. Because Γ is empty, the "for all" premise in Theorem 1 is vacuously true. □

Theorem 1 demonstrates that a variety of useful constructs are logically consistent. For instance, we can add a lattice structure to FLAFOL's labels. We can define join ($\sqcup$) and meet ($\sqcap$) as binary function symbols on labels and $\top$ and $\bot$ as label constants. Then we can simply place the lattice axioms (e.g., $\forall \ell : \textsf{Label}. \ \ell \sqsubseteq \top$) in our context to achieve the desired result. Since none of the lattice axioms include False, Theorem 1 ensures that they are consistent additions to the logic.

5.2 Simulation

Most modal logics give meaning to the modality via a rule of the following form, where $p \ \textsf{\textbf{says}}_\ell \ \Gamma$ refers to the context Γ where every formula in Γ has $p \ \textsf{\textbf{says}}_\ell$ in front:

$$\frac{\Gamma \vdash \varphi}{p \ \textsf{\textbf{says}}_\ell \ \Gamma \vdash p \ \textsf{\textbf{says}}_\ell \ \varphi}$$

This rule says that every principal is a perfect reasoner, in the sense that they always reason correctly from their beliefs. However, it also traps the reasoning of a principal "in its own head," so that communication is not possible when reasoning about a principal.

By introducing generalized principals and including left and right **says** rules, FLAFOL allows for communication, even under **says** statements. However, FLAFOL principals are also intended to be perfect

$$\text{CSCRefl}\ \frac{}{\Gamma \ll \Gamma \vdash \varphi \mathbin{@} g} \qquad \text{CSCUnion}\ \frac{\Delta_1 \ll \Gamma \vdash \varphi \mathbin{@} g \qquad \Delta_2 \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta_1 \cup \Delta_2 \ll \Gamma \vdash \varphi \mathbin{@} g}$$

$$\text{CSCOrL1}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \text{CSCImpR}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \to \psi \mathbin{@} g}$$

Figure 5: Selected Rules for Compatible Supercontexts

reasoners, so we would thus like to prove that the "normal" rule above is admissible. Because FLAFOL interprets the premises of implications as ground truth, we need to change the rule to put appropriate **says** statements in front of premises of implications. Appendix C provides details of this transformation and the complete theorem statement.

5.3 Compatible Supercontexts

To prove Theorem 1 we needed to consider the possible locations of *formulae* within a sequent, but in Section 6 we will need to reason about the possible locations of *beliefs*. To enable this, we introduce the concept of a *compatible supercontext* (CSC). Informally, the CSCs of a sequent are those contexts that contain all of the information in the current context, along with any counterfactual information that can be considered during a proof. Intuitively, the rules OrL and ImpL allow a generalized principal to consider such information by using either side of a disjunction or the conclusion of an implication. If it is possible to consider such a counterfactual, there is a CSC which contains it. We use the syntax $\Delta \ll \Gamma \vdash \varphi \mathbin{@} g$ to denote that Δ is a CSC of the sequent $\Gamma \vdash \varphi \mathbin{@} g$. Figure 5 contains selected rules for CSCs. Others can be found in Appendix G.

Since all of the information in Γ has already been discovered by the generalized principal who believes that information, we require that $\Gamma \ll \Gamma \vdash \varphi \mathbin{@} g$ with CSCRefl.

If we can discover two sets of information, we can discover everything in the union of those sets using CSCUnion. This rule feels different from the others, since it axiomatizes certain *properties* of CSCs. We conjecture that there is an alternative presentation of CSCs where we can prove this rule.

The rest of the rules for CSCs essentially follow the proof rules, so that any belief added to the context during a proof can be added to a CSC. For instance CSCOrL1 and CSCOrL2 allow either branch of an assumed disjunction to be added to a CSC, following the two branches of the OrL rule of FLAFOL.

If a context appears in a proof of a sequent, then it is a CSC of that sequent. We refer to this as the *compatible-supercontext property* (CSC property).

Theorem 2 (CSC Property)**.** *If* $\Delta \vdash \psi \mathbin{@} g'$ *appears in a proof of* $\Gamma \vdash \varphi \mathbin{@} g$*, then* $\Delta \ll \Gamma \vdash \varphi \mathbin{@} g$*.*

5.4 Cut Elimination

In constructing a proof, it is often useful to create a lemma, prove it separately, and use it in the main proof. If we both prove and use the lemma in the same context, the main proof follows in that context as well. We can formalize this via the following rule:

$$\text{Cut}\ \frac{\Gamma \vdash \varphi \mathbin{@} g_1 \qquad \Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g_2}{\Gamma \vdash \psi \mathbin{@} g_2}$$

This rule is enormously powerful. It allows us to not only create lemmata to use in a proof, but also prove things that do not obviously have other proofs. For instance, consider the rule

$$\textsc{UnsaysR}\ \frac{\Gamma \vdash p\ \textsf{\textbf{says}}_\ell\ \varphi\ @\ g}{\Gamma \vdash \varphi\ @\ g \cdot p\langle \ell \rangle}$$

We can show that this rule is *admissible*—meaning any sequent provable with this rule is provable without it—by cutting a proof of the sequent $\Gamma \vdash p\ \textsf{\textbf{says}}_\ell\ \varphi\ @\ g$ with the following proof:[7]

$$\textsc{SaysL}\ \frac{\textsc{Ax}\ \overline{\varphi\ @\ g \cdot p\langle \ell \rangle \vdash \varphi\ @\ g \cdot p\langle \ell \rangle}}{p\ \textsf{\textbf{says}}_\ell\ \varphi\ @\ g \vdash \varphi\ @\ g \cdot p\langle \ell \rangle}$$

However, the CUT rule allows an arbitrary formula to appear on both sides of the turnstile in a proof. That formula may not even be a subformula of anything in the sequent at the root of the proof-tree! This would seemingly destroy the CSC property that FLAFOL enjoys, and which we rely on in order to prove FLAFOL's security results. As is standard in sequent calculus proof theory, we show that CUT can be admitted, allowing FLAFOL the proof power of CUT while maintaining the analytic power of the CSC property.

Theorem 3 (Cut Elimination)**.** *The* CUT *rule is admissible.*

To prove Theorem 3, we first normalize each FLAFOL proof and then induct on the formula φ followed by each proof in turn. Both of these inductions are very involved. Appendix D contains more details.

This theorem is one of the key theorems of proof theory [17, 40]. Frank Pfenning has called it "[t]he central property of sequent calculi" [33]. From the propositions-as-types perspective, cut elimination is preservation of types under substitution.

5.5 Implications and Communication

Recall from Section 4 how we interpret implication formulae such as Alice $\textsf{\textbf{says}}_\ell\ (\varphi \to \psi)$: if φ is true about the system, then Alice knows ψ at label ℓ. We can now understand why we use this interpretation.

Imagine we replace rules IMPL and IMPR with rules that interpret the above formula in a more intuitive fashion: if Alice believes φ at label ℓ, then she also believes ψ at ℓ. This would allow us to prove that **says** distributes over implication: if Alice $\textsf{\textbf{says}}_\ell\ (\varphi \to \psi)$ then (Alice $\textsf{\textbf{says}}_\ell\ \varphi$) $\to$ (Alice $\textsf{\textbf{says}}_\ell\ \psi$). It would also, unfortunately, allow us to prove the converse.

In this setting we can provide a proof that contains a cut we cannot eliminate and demonstrates a security flaw. Imagine that Alice receives a TopSecret version of φ from Cathy, and she wants to prove ψ at TopSecret. Alice can also prove ψ publicly if she believes φ privately, but doing so requires sending φ to Bob. By packaging this proof into an implication using these new rules and then using variance, we obtain a proof that, if Alice believes φ at TopSecret, she can prove ψ at TopSecret.[8] We can cut these two proofs together, but eliminating this cut would force Alice or Cathy to send a TopSecret belief to Bob, which they are unwilling to do. Appendix E contains details of this example.

The insecurity above stems from the ability to both distribute and *un-distribute* **says** across implications while also using the variance and forward rules. Adopting a propositions-as-types viewpoint provides further insight. In this perspective, the **says** modalities are type constructors, the variance and forwarding rules act as subtyping relations on the resulting types, and implications are functions. The forward and variance rules require functions to behave *contravariantly* on their inputs, as normal. Also allowing **says** to distribute over implications in both directions, however, would force functions to behave *covariantly* on their inputs.

[7]Not only can UNSAYSR be proven without CUT (as can all FLAFOL proofs), it is actually important for proving cut elimination. See the Coq code.

[8]Note that IMPL and IMPR do not allow this proof.

A standard type-theoretic argument suggests that this is incoherent, and it makes β-reduction—which corresponds to cut elimination—impossible. By treating premises as ground truth, functions become *invariant* on their premises, allowing us to prove cut elimination for FLAFOL.

6 Non-Interference

Both authorization logics and information flow systems have important security properties called *non-interference* [15, 16, 18]. On the face, these two notions of non-interference look very different, but their core intuitions are the same. Both statements aim to prevent one belief or piece of data from interfering with another—*even indirectly*—unless the security policies permit an influence. Authorization logics traditionally define trust relationships between principals and non-interference requires that p's beliefs affect the provability of q's beliefs only when q trusts p. Information flow control systems generally specify policies as labels on program data and use the label flows-to relation to constrain how inputs can affect outputs. For non-interference to hold, changing an input with label ℓ_1 can only alter an output with label ℓ_2 if $\ell_1 \sqsubseteq \ell_2$.

FLAFOL views both trust between principals and flows between labels as ways to constrain communication of beliefs. The forward rules model an authorization-logic-style sending of beliefs from one principal to another based on their trust relationships. The label variance rules model a single principal transferring beliefs between labels based on the flow relationship between them. By reasoning about generalized principals, which include both the principal and the label, we are able to capture both at the same time. The result (Theorem 5) mirrors the structure of existing authorization logic non-interference statements [1, 16]. No similar theorem reasons about information flow or applies to policies combining discoverable trust and logical disjunction. Theorem 5 does both.

6.1 Trust in FLAFOL

Building a notion of trust on generalized principals requires us to consider both the trust of the underlying (regular) principals and label flows. The explicit label flow relation ($\sqsubseteq$) cleanly captures restrictions on changing labels. Trust between principals requires more care. Alice may trust Bob with public data, but that does not mean she trusts him with secret data. Similarly, Alice may believe that Bob can influence low integrity data without believing Bob is authorized to influence high integrity data. This need to trust principals differently at different labels leads us to define our trust in terms of the two permission relations: $\mathsf{CanRead}(p, \ell)$ and $\mathsf{CanWrite}(p, \ell)$.

We group label flows and principal trust together in a meta-level statement relating generalized principals. As this relation is the fundamental notion of trust in FLAFOL, we follow existing authorization logic literature and call it *speaks for*.

The speaks-for relation captures any way that one generalized principal's beliefs can be safely transfered to another. This can happen through flow relationships ($g \cdot p\langle\ell\rangle$ speaks for $g \cdot p\langle\ell'\rangle$ if $\ell \sqsubseteq \ell'$), forwarding ($g \cdot p\langle\ell\rangle$ speaks for $g \cdot q\langle\ell\rangle$ if p can forward beliefs at ℓ to q), and introspection ($g \cdot p\langle\ell\rangle$ speaks for $g \cdot p\langle\ell\rangle \cdot p\langle\ell\rangle$ and vice versa). We formalize speaks-for with the rules in Figure 6.

To validate this notion of trust, we note that existing authorization logics often define speaks-for as an atomic relation and create trust by requiring that, if p speaks for q, then p's beliefs can be transfered to q. As our speaks-for relation exactly mirrors FLAFOL's rules for communication, it enjoys this same property.

Theorem 4 (Speaks-For Elimination)**.** *The following rule is admissible in FLAFOL:*

$$\textsc{ElimSF}\ \frac{\Gamma \vdash \varphi \mathbin{@} g_1 \qquad \Gamma \vdash g_1 \mathsf{\ SF\ } g_2}{\Gamma \vdash \varphi \mathbin{@} g_2}$$

$$\textsc{ReflSF}\ \frac{}{\Gamma \vdash g \ \mathsf{SF}\ g} \qquad \textsc{ExtSF}\ \frac{\Gamma \vdash g_1 \ \mathsf{SF}\ g_2}{\Gamma \vdash g_1 \cdot p\langle \ell \rangle \ \mathsf{SF}\ g_2 \cdot p\langle \ell \rangle}$$

$$\textsc{SelfLSF}\ \frac{}{\Gamma \vdash g \cdot p\langle \ell \rangle \ \mathsf{SF}\ g \cdot p\langle \ell \rangle \cdot p\langle \ell \rangle} \qquad \textsc{SelfRSF}\ \frac{}{\Gamma \vdash g \cdot p\langle \ell \rangle \cdot p\langle \ell \rangle \ \mathsf{SF}\ g \cdot p\langle \ell \rangle}$$

$$\textsc{VarSF}\ \frac{\Gamma \vdash \ell \sqsubseteq \ell' \ @\ g \cdot p\langle \ell' \rangle}{\Gamma \vdash g \cdot p\langle \ell \rangle \ \mathsf{SF}\ g \cdot p\langle \ell' \rangle} \qquad \textsc{FwdSF}\ \frac{\Gamma \vdash \mathsf{CanRead}(q, \ell) \ @\ g \cdot p\langle \ell \rangle \qquad \Gamma \vdash \mathsf{CanWrite}(p, \ell) \ @\ g \cdot q\langle \ell \rangle}{\Gamma \vdash g \cdot p\langle \ell \rangle \ \mathsf{SF}\ g \cdot q\langle \ell \rangle}$$

$$\textsc{TransSF}\ \frac{\Gamma \vdash g_1 \ \mathsf{SF}\ g_2 \qquad \Gamma \vdash g_2 \ \mathsf{SF}\ g_3}{\Gamma \vdash g_1 \ \mathsf{SF}\ g_3}$$

Figure 6: The rules defining speaks for.

This notion of trust allows us to begin structuring a non-interference statement. We might like to say that beliefs of g_1 can only influence beliefs of g_2 if $\Gamma \vdash g_1 \ \mathsf{SF}\ g_2$. Formally, this might take the form: if $\Gamma, (\varphi \ @\ g_1) \vdash \psi \ @\ g_2$ is provable, then either $\Gamma \vdash \psi \ @\ g_2$ is provable or $\Gamma \vdash g_1 \ \mathsf{SF}\ g_2$. Unfortunately, this statement is false for three critical reasons: **says** statements, implication, and the combination of discoverable trust and disjunctions.

6.2 Says Statements and Non-Interference

The first way to break the proposed non-interference statement above is simply by moving affirmations of a statement between the formula—using **says**—and the generalized principal who believes it. For example, we can trivially prove $p \ \textbf{says}_\ell\ \varphi \ @\ \langle\rangle \vdash \varphi \ @\ \langle\rangle \cdot p\langle \ell \rangle$, yet we cannot prove $\langle\rangle \ \mathsf{SF}\ \langle\rangle \cdot p\langle \ell \rangle$.

To address this case, we can view $p \ \textbf{says}_\ell\ \varphi \ @\ \langle\rangle$ as a statement that $\langle\rangle \cdot p\langle \ell \rangle$ believes φ. This insight suggests that we might generally push all **says** modalities into the generalized principal. We can do this for simple formulae, but the process breaks down with conjunction and disjunction. In those cases, the different sides may have different **says** modalities, and either side could influence a belief through the different resulting generalized principals. We alleviate this concern by considering a *set* of generalized principals referenced in a given belief. We build this set using an operator $\mathcal{G}$:

$$\mathcal{G}(\chi \ @\ g) \triangleq \begin{cases} \mathcal{G}(\varphi \ @\ g \cdot p\langle \ell \rangle) & \chi = p \ \textbf{says}_\ell\ \varphi \\ \mathcal{G}(\varphi \ @\ g) \cup \mathcal{G}(\psi \ @\ g) & \chi = \varphi \wedge \psi \text{ or } \varphi \vee \psi \\ \mathcal{G}(\psi \ @\ g) & \chi = \varphi \rightarrow \psi \\ \bigcup_{t:\sigma} \mathcal{G}(\varphi[x \mapsto t] \ @\ g) & \chi = \forall x{:}\sigma.\, \varphi \text{ or } \exists x{:}\sigma.\, \varphi \\ \{g\} & \text{otherwise} \end{cases}$$

For implications, $\mathcal{G}$ only considers the consequent, as the implication cannot affect the provability of a belief unless its consequent can. For quantified formulae, a proof may substitute any term of the correct sort for the bound variable, so we must as well.

Using this new operator, we can patch the hole **says** statements created in our previous non-interference statement, producing the following: If $\Gamma, (\varphi \ @\ g_1) \vdash \psi \ @\ g_2$, then either $\Gamma \vdash \psi \ @\ g_2$, or there is some $g_1' \in \mathcal{G}(\varphi \ @\ g_1)$, $g_2' \in \mathcal{G}(\psi \ @\ g_2)$, and some g_1'' such that $\Gamma \vdash g_1' \cdot g_1'' \ \mathsf{SF}\ g_2'$.

Here g_1'' represents the ability of a generalized principal to ship entire simulations to other generalized principals. In particular, the forward and variance rules operate on an "active" prefix of the current generalized principal; g_1'' represents the "inactive" suffix.

$$\text{SF-CI}\ \frac{\Gamma \vdash g_1\ \mathsf{SF}\ g_2}{\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_2} \qquad \text{EXTCI}\ \frac{\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_2}{\Gamma \vdash g_1 \cdot g'\ \mathsf{CanInfl}\ g_2 \cdot g'}$$

$$\text{TRANSCI}\ \frac{\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_2 \qquad \Gamma \vdash g_2\ \mathsf{CanInfl}\ g_3}{\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_3}$$

$$\text{IMPCI}\ \frac{\varphi \to \psi \mathbin{@} g \in \Gamma \qquad g_1 \in \mathcal{G}(\varphi \mathbin{@} \langle\rangle) \qquad g_2 \in \mathcal{G}(\psi \mathbin{@} g)}{\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_2}$$

Figure 7: The rules defining the *can influence* relation.

The $\mathcal{G}$ operator converts reasoning about beliefs from the object level (FLAFOL formulae) to the meta level (generalized principals). FLAFOL's ability to freely move between the two forces us to push all such reasoning in the same direction to effectively compare the reasoner in two different beliefs. Prior authorization logics do not contain a meta-level version of **says**, meaning similar conversions do not even make sense.

6.3 Implications

While use of the $\mathcal{G}$ function solves part of the problem with our original non-interference proposal, it does not address all of the problems. Implications can implicitly create new trust relationships, allowing beliefs of one generalized principal to affect beliefs of another, even when no speaks-for relationship exists. To understand how this can occur, we revisit our example of preventing SQL injection attacks from from Section 2.2.

Recall from Section 2.2 that a web server might treat sanitized versions of low-integrity input as high integrity. Further recall, it might represent this willingness with the following implication.

$$\mathsf{System}\ \mathbf{says}_{\mathsf{LInt}}\ \mathsf{DBInput}(x) \to \mathsf{System}\ \mathbf{says}_{\mathsf{HInt}}\ \mathsf{DBInput}(\mathsf{San}(x))$$

In an intuitively-sensible context where System believes $\mathsf{HInt} \sqsubseteq \mathsf{LInt}$—high integrity flows to low integrity—but not vice versa, there is no way to prove $\mathsf{System}\langle\mathsf{LInt}\rangle\ \mathsf{SF}\ \mathsf{System}\langle\mathsf{HInt}\rangle$. The presence of this implication, however, allows some beliefs at $\mathsf{System}\langle\mathsf{LInt}\rangle$ to influence beliefs at $\mathsf{System}\langle\mathsf{HInt}\rangle$. This influence is actually an endorsement from LInt to HInt, and our speaks-for relation explicitly does not capture such effects.

Prior work manages this trust-creating effect of implications either by claiming security only when all implications are provable [1] or by explicitly using assumed implications to represent trust [16]. We hew closer to the latter model and make the implicit trust of implications explicit in our statement of non-interference. We therefore cannot use the speaks-for relation, so we construct a new relation between generalized principals we call *can influence*.

Intuitively, g_1 can influence g_2—denoted $\Gamma \vdash g_1\ \mathsf{CanInfl}\ g_2$—if either g_1 speaks for g_2 or there is an implication in Γ that allows a belief of g_1 to affect the provability of a belief of g_2. This relation, formally defined in Figure 7, uses the $\mathcal{G}$ operator discussed above to capture the generalized principals actually discussed by each subformla of the implication. Because FLAFOL interprets the premise of an implication as a condition whose modality is independent of the entire belief, so too does the can-influence relation. The relation is also transitive, allowing it to capture the fact that a proof may require many steps to go from a belief at g_1 to a belief at g_2.

Simply taking our attempted non-interference statement from above and replacing speaks-for with can-influence allows us to straightforwardly capture the effect of implications on trust within the system.

While this change may appear small, it results in a highly conservative estimate of possible influence. Implications are precise statements that can allow usually-disallowed information flows under very particular circumstances. Unfortunately, because our non-interference statement only considers the generalized principals involved, not the entire beliefs, it cannot represent the same level of precision. A single precise implication added to a context can therefore relate whole classes of previously-unrelated generalized principals, eliminating the ability for non-interference to say anything about their relative security.

This same lack of precision in information flow non-interference statements has resulted in long lines of research on how to precisely model or safely restrict declassification and endorsement [6, 12, 14, 25, 27, 32, 35, 36, 42, 44]. We believe it would be interesting future work to apply these analyses and restrictions to FLAFOL to produce more precise statements of security.

6.4 Discovering Trust with Disjunctions

The $\mathcal{G}$ operator and can-influence relation address difficulties from both **says** formulae and implications, but our statement of non-interference still does not account for the combination of disjunctions and the ability to discover trust relationships. To understand the effect of these two features in combination, recall the reinsurance example from Section 2.3. Bob can derive $\mathsf{CanWrite}(I_1, \ell_H)$ if he already believes both $\mathsf{CanWrite}(I_1, \ell_H) \vee \mathsf{CanWrite}(I_2, \ell_H)$ and I_2 $\mathbf{says}_{\ell_H}$ $\mathsf{CanWrite}(I_1, \ell_H)$.

We clearly cannot remove either of Bob's beliefs and still prove the result. Our desired theorem statement would thus require that $\mathsf{Bob}\langle \ell_H \rangle \cdot I_2\langle \ell_H \rangle$ can influence $\mathsf{Bob}\langle \ell_H \rangle$, which there is no way to prove. The reason the sequent is still provable, as we noted in Section 2.3, is that Bob can *discover* trust in I_2 when he branches on an Or statement, which then allows I_2 to influence Bob. In this branch, we can prove $\mathsf{Bob}\langle \ell_H \rangle \cdot I_2\langle \ell_H \rangle$ SF $\mathsf{Bob}\langle \ell_H \rangle \cdot \mathsf{Bob}\langle \ell_H \rangle$, which then speaks for $\mathsf{Bob}\langle \ell_H \rangle$.

To handle such assumptions, we cannot simply consider the context in which we are proving a sequent; we must consider any context that can appear in the proof of that sequent. We developed the notion of compatible supercontexts in Section 5.3 for exactly this purpose. Indeed, if we replace Γ with an appropriate CSC when checking the potential influence of generalized principals, we remove the last barrier to a true non-interference theorem.

6.5 Formal Non-Interference

The techniques above allow us to modify our attempted non-interference statement into a theorem that holds.

Theorem 5 (Non-Interference). *For all contexts Γ and beliefs φ @ g_1 and ψ @ g_2, if*

$$\Gamma, \varphi \ @\ g_1 \vdash \psi \ @\ g_2,$$

then either (1) $\Gamma \vdash \psi$ @ g_2, or (2) there is some $\Delta \ll \Gamma, \varphi$ @ $g_1 \vdash \psi$ @ g_2, $g_1' \in \mathcal{G}(\varphi \ @\ g_1)$, $g_2' \in \mathcal{G}(\psi \ @\ g_2)$, and g_1'' such that $\Delta \vdash g_1' \cdot g_1''$ $\mathsf{CanInfl}$ g_2'.

The proof of this theorem follows by induction on the proof of Γ, φ @ $g_1 \vdash \psi$ @ g_2. For each proof rule, we argue that either φ @ g_1 is unnecessary for all premises or we can extend an influence from one or more subproofs to an influence from φ @ g_1 to ψ @ g_2.

This theorem limits when a belief φ @ g_1 can be necessary to prove ψ @ g_2 in context Γ, much like other authorization logic non-interference statements [1, 16]. As we mentioned above, however, it is the first such non-interference statement for any authorization logic supporting all first-order connectives and discoverable trust. Moreover, it describes how FLAFOL mitigates both:

- communication between principals, through $\mathsf{CanRead}$ and $\mathsf{CanWrite}$ statements, and

- movement of information between security levels represented by information flow labels, via flows-to statements.

The CanInfl relation seems to make our non-interference statement much less precise than we would like. After all, implications precisely specify what beliefs can be declassified or endorsed, whereas CanInfl conservatively assumes any beliefs can move between the relevant generalized principals. This lack of precision serves a purpose. It allows us to reason about any implications, including those that arbitrarily change principals and labels, something which other no authorization logics have done before. It is therefore worth noting that, when all of the implications in the context are provable, the theorem holds *even if you replace* CanInfl *with* SF *everywhere.* The same proof works, with some simple repair in the IMPL case.

Another complaint of imprecision applies to compatible supercontexts. Specifically, if any principal assumes $\varphi \vee \neg\varphi$ for any formula φ, then there is a CSC in which that principal has assumed both, even though these are arrived at through mutually-exclusive choices. Since CSCs have been added in order to allow disjunctions and discoverable trust to co-exist, it is good to know that if we disallow either, CSCs are not required for non-interference. That is, if there are no disjunctions in the context, then we can always instantiate the Δ in Theorem 5 with $\Gamma, \varphi \mathbin{@} g_1$. Similarly, if every permission that is provable in any CSC of $\Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g_2$ is provable under $\Gamma, \varphi \mathbin{@} g_1$, then we can again always instantiate Δ with $\Gamma, \varphi \mathbin{@} g_1$.

Together, these points demonstrate that there are only two types of poorly-behaved formulae that force the imprecision in Theorem 5. This further shows that our non-interference result is no less precise than those of other authorization logics in the absence of such formulae. We add imprecision only when needed to allow our statement to apply to more proofs. Interesting future research would allow for a more precise non-interference theorem even in the presence of such formulae.

To see how Theorem 5 corresponds to traditional non-interference results for information flow, consider a setting where every principal agrees on the same label ordering, and where there are no implications corresponding to declassifications or endorsements. Then any two contexts Γ and Γ' which disagree only on beliefs labeled above some ℓ can prove exactly the same things at label ℓ—$\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle$ if and only if $\Gamma' \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle$—since Theorem 5 allows us to delete all of the beliefs on which they disagree. If we view contexts as inputs, as in a propositions-as-types interpretation, then this says that changing high inputs cannot change low results.

7 Future Work

FLAFOL is already very powerful, but it suggests numerous avenues for future work.

First, FLAFOL only disallows *direct* flows of information in proofs, but checking proofs can cause communication and potentially leak information. Importantly, eliminating cuts in proofs can *increase* the information leaked during proof-checking because eliminating cuts can reduce the uncertainty about which discoveries can be made during a proof. This is disturbing, since we would like to be able to perform sound security analyses on proofs with cut; system designers should not need to understand the very complicated cut-elimination proof. The *program counter* mechanism used by information flow control systems like Fabric [26] and FLAM [4] seems to prevent similar leaks. Incorporating program counter labels to limit communication in FLAFOL proofs could eliminate these leaks in FLAFOL as well.

This improvement also widens the range of programs that can safely use FLAFOL. Justifications for authorization need to be found as well as checked. From the point of view of an authorization logic, this corresponds to proof search. Searching for an authorization proof in a distributed system, however, may require communication between principals, potentially leaking why they are searching for this proof in the first place. One avenue forward embeds FLAFOL in a language with information-flow types, and runs proof search in that language. This would guarantee that the proof search does not leak data assuming FLAFOL proofs do not leak data when checked.

We have developed new techniques to reason about authorization-logic proofs in order to prove non-interference for FLAFOL. These reasoning principles could be expanded and used in other logics. For instance, using the tools developed in Section 6, we should be able to give non-interference proofs for logics like NAL [37] and FOCAL [20] which reason about implication and disjunction. We should also be able to add disjunction and implication to logics like DCC [1, 2] while still providing a non-interference theorem.

Another avenue of further work would understand better how **says** statements can interact with other logical connectives. For instance, one might want to model a principal who cannot observe whether they are holding evidence of φ or of ψ. For instance, we might want to model a principal p who receives an encrypted message containing a bit b. Then p knows that either $b = 0$ or $b = 1$, but p cannot examine the evidence to determine which. Thus, while p $\textbf{says}_\ell$ $(b = 0 \vee b = 1)$, we should not be able to show that $(p\ \textbf{says}_\ell\ b = 0) \vee (p\ \textbf{says}_\ell\ b = 1)$. A NuPRL-like "squash" operator, which prevents evidence from being used [11], could model this, but further research is needed for FLAFOL to reason about the security of such protocols.

A similar avenue for future work involves exploring ways to allow **says** to distribute over implications while remaining coherent. One potential approach would be to confine most reasoning to a single generalized principal, but this would restrict implications so that the principal who believes them cannot communicate in their proof. The consequences of such a restriction on modeling real-world systems are unclear.

Finally, it would be nice to reason about the *temporal* components of authorization; this is one place where work on information flow far outstrips that on authorization logic [6, 12, 14, 25, 27, 32, 35, 36, 42, 44]. Trust relationships may change over time, allowing or disallowing communication pathways. Understanding how this changes which authorizations should be provable, and how this affects information-flow policies, is a rich area for exploration.

8 Related Work

Prior work in both information flow control and authorization logics has explored connections between authorization and information security. The Decentralized Label Model [30] incorporates a notion of ownership into its information flow policies, specifying who may authorize exceptions to a policy.

The Flow-Limited Authorization Model (FLAM) [4] was the first information-flow label model to directly consider the confidentiality and integrity of policies when authorizing information flows. Prior work on Rx [39] and RTI [7] enforced information flow policies via *roles* whose membership are protected with confidentiality and integrity labels.

We deviate from these works in several important ways. First, FLAFOL is a formal authorization logic. Second, we employ both principals and labels, but keep them entirely separate. Many information flow models are defined with respect to an abstract security lattice and omit any direct representation of principals. The Decentralized Label Model [31] expresses labels in terms of principals. FLAM [4] takes this a step further and represents principals directly as a combination of confidentiality and integrity labels. This view restricts FLAM from reasoning about labels with policies other than confidentiality and integrity, since they might necessitate subtle changes to FLAM's reasoning rules.

Unifying principals and labels also undermines FLAM's effectiveness as an authorization logic. It is convenient to construct complex policies from simpler ones, such as a policy protecting Alice's confidentiality and Bob's integrity. FLAM regards such a compound policy as a principal, but this principal does not represent an actual entity in the system. These principals break the connection between principals and system entities often present in authorization logics. While it is certainly possible to represent these in FLAFOL, FLAFOL does not necessarily force a reasoner to break this connection between principals and system entities.

Becker [9] explores preventing probing attacks, authorization queries which leak secret information, in Datalog-based authorization logics like DKAL [19] and SecPAL [10]. In SecPAL+ [8], Becker proposes a new *can listen to* operator, similar to FLAFOL's CanRead permission, that expresses who is permitted to learn specific statements. However, *can listen to* expresses permissions on specific statements, not labels as CanRead does. Moreover, FLAFOL tracks dependencies between statements using these labels, so the security consequences of adding a new permission are more explicit.

Garg and Pfenning [16] present an authorization logic and a non-interference result that ensures untrusted principals cannot influence the truth of statements made by other principals. FLAFOL differs from this logic in two ways. First, FLAFOL supports all first-order connectives while Garg and Pfenning only support implication and universal quantification. Second, Garg and Pfenning only use implications to encode trust, rather than having an explicit trust relation between principals.

The Dependency Core Calculus [1, 2] (DCC) has been used to model both information flow control and authorization, but not at the same time. DCC also has a non-interference property, but like many authorization logics, it employs an external lattice to express trust between principals. FLAFOL supports both finer-grained trust and discoverable trust.

The Flow-Limited Authorization Calculus [5] uses ideas from FLAM and DCC to support discoverable trust. FLAC and Polymorphic DCC [1] are based on System F, which contains some elements of second-order logic since it supports universal quantification over types, but does not support some features of first-order logic like existential quantification.

Finally, AURA [22, 23] embeds DCC into a language with dependent types in order to explore how authorization logic interacts with programs. Their non-interference result for authorization comes directly from DCC, but they express first-order properties by combining constructs from the programming language with constructs from DCC. This makes it unclear what guarantees the theorem provides. Jia and Zdancewic encode information-flow labels into AURA as principals and develop a non-interference theorem in the style of information-flow systems [23]. This setup unfortunately makes it impossible for principals to disagree about the meaning of labels, since the labels themselves define their properties.

9 Conclusion

We have introduced FLAFOL, a first-order logic which combines notions of trust from both authorization and information flow. It also provides a concrete model of communication that respects this combination. Furthermore, FLAFOL gives principals the ability to reason about each other's differing opinions, including differing opinions about trust. FLAFOL has a powerful non-interference theorem that navigates this complexity, a top-tier result for authorization logics. It is, moreover, the most complete first-order logic with such a guarantee.

Acknowledgments

We would like to thank Andrew Myers and Jed Liu for early feedback on the design of FLAFOL. Discussion with Deepak Garg gave us insight into how to introduce FLAFOL, while Phokion G. Kolatis pointed us to some related work. We would futhermore like to thank Cosku Acay, Arthur Azevedo de Amorim, Dietrich Geisler, Elisavet Kozyri, and Matthew Milano for their valuable help editing this paper.

Funding for this work was provided by NSF grant #1704788, NSF CAREER grant #1750060, and a National Defense Science and Engineering Graduate Fellowship. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect those of these sponsors.

References

[1] M. Abadi. Access control in a core calculus of dependency. In *11th ACM SIGPLAN Int'l Conf. on Functional Programming*, pages 263–273, New York, NY, USA, 2006. ACM.

[2] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke. A core calculus of dependency. In *26th ACM Symp. on Principles of Programming Languages (POPL)*, pages 147–160, Jan. 1999.

[3] M. Algehed. Short paper: A perspective on the dependency core calculus. *PLAS*, October 2018.

[4] O. Arden, J. Liu, and A. C. Myers. Flow-limited authorization. In *28th IEEE Symp. on Computer Security Foundations (CSF)*, pages 569–583, July 2015.

[5] O. Arden and A. C. Myers. A calculus for flow-limited authorization. In *29th IEEE Symp. on Computer Security Foundations (CSF)*, pages 135–147, June 2016.

[6] A. Askarov and A. C. Myers. Attacker control and impact for confidentiality and integrity. *Logical Methods in Computer Science*, 7(3), Sept. 2011.

[7] S. Bandhakavi, W. Winsborough, and M. Winslett. A trust management approach for flexible policy management in security-typed languages. In *Computer Security Foundations Symposium, 2008*, pages 33–47, 2008.

[8] M. Y. Becker. Information flow in credential systems. In *23rd IEEE Symp. on Computer Security Foundations (CSF)*, pages 171–185. IEEE, 2010.

[9] M. Y. Becker. Information flow in trust management systems. *Journal of Computer Security*, 20(6):677–708, 2012.

[10] M. Y. Becker, C. Fournet, and A. D. Gordon. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security*, 18(4):619–665, 2010.

[11] J. L. Caldwell. Classical propositional decidability via nuprl proof extraction. *TPHOLs*, 1998.

[12] E. Cecchetti, A. C. Myers, and O. Arden. Nonmalleable information flow control. In *24th ACM Conf. on Computer and Communications Security (CCS)*, pages 1875–1891, Oct. 2017.

[13] A. Charguéraud. The locally nameless representation. *Jounal of Automated Reasoning*, 49(3):363–408, October 2012.

[14] S. Chong and A. C. Myers. End-to-end enforcement of erasure and declassification. In *IEEE Symp. on Computer Security Foundations (CSF)*, pages 98–111, June 2008.

[15] D. E. Denning. A lattice model of secure information flow. *Comm. of the ACM*, 19(5):236–243, 1976.

[16] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In *19th IEEE Computer Security Foundations Workshop (CSFW)*, New Jersey, USA, 2006. IEEE.

[17] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[18] J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symp. on Security and Privacy*, pages 11–20, Apr. 1982.

[19] Y. Gurevich and I. Neeman. DKAL: Distributed-knowledge authorization language. In *IEEE Symp. on Computer Security Foundations (CSF)*, pages 149–162. IEEE, 2008.

[20] A. K. Hirsch and M. R. Clarkson. Belief semantics of authorization logic. *CCS*, pages 561–572, November 2013.

[21] J. Howell and D. Kotz. A formal semantics for SPKI. In *ESORICS 2000*, volume 1895 of *Lecture Notes in Computer Science*, pages 140–158. Springer Berlin Heidelberg, 2000.

[22] L. Jia, J. A. Vaughan, K. Mazurak, J. Zhao, L. Zarko, J. Schorr, and S. Zdancewic. AURA: A programming language for authorization and audit. In *13th ACM SIGPLAN Int'l Conf. on Functional Programming*, Sept. 2008.

[23] L. Jia and S. Zdancewic. Encoding information flow in AURA. *PLAS*, pages 17–29, June 2009.

[24] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. In *13th ACM Symp. on Operating System Principles (SOSP)*, pages 165–182, Oct. 1991.

[25] P. Li and S. Zdancewic. Downgrading policies and relaxed noninterference. In *32nd ACM Symp. on Principles of Programming Languages (POPL)*, Long Beach, CA, Jan. 2005.

[26] J. Liu, O. Arden, M. D. George, and A. C. Myers. Fabric: Building open distributed systems securely by construction. *J. Computer Security*, 25(4–5):319–321, May 2017.

[27] H. Mantel and D. Sands. Controlled Declassification based on Intransitive Noninterference. In *2nd ASIAN Symposium on Programming Languages and Systems, APLAS 2004*, LNCS 3303, pages 129–145, Taipei, Taiwan, Nov. 2004. Springer-Verlag.

[28] The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2004. Version 8.0.

[29] M. P. Milano and A. C. Myers. MixT: A language for mixing consistency in geodistributed transactions. In *39th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)*, June 2018.

[30] A. C. Myers and B. Liskov. Complete, safe information flow with decentralized labels. In *IEEE Symp. on Security and Privacy*, pages 186–197, May 1998.

[31] A. C. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410–442, Oct. 2000.

[32] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security*, 14(2):157–196, 2006.

[33] F. Pfenning. Structural cut elimination. *LICS*, pages 156–166, June 1995.

[34] V. Rajani, D. Garg, and T. Rezk. On access control, capabilities, their equivalence, and confused deputy attacks. *CSF*, June 2016.

[35] A. Sabelfeld and A. C. Myers. A model for delimited release. In *2003 International Symposium on Software Security*, number 3233 in Lecture Notes in Computer Science, pages 174–191. Springer-Verlag, 2004.

[36] A. Sabelfeld and D. Sands. Dimensions and principles of declassification. In *18th IEEE Computer Security Foundations Workshop (CSFW)*, pages 255–269, June 2005.

[37] F. B. Schneider, K. Walsh, and E. G. Sirer. Nexus Authorization Logic (NAL): Design rationale and applications. *ACM Trans. Inf. Syst. Secur.*, 14(1):8:1–8:28, June 2011.

[38] E. G. Sirer, W. D. Bruijin, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider. Logical attestation: An authorization architecture for trustworthy computing. In *11th ACM Symp. on Operating System Principles (SOSP)*, 2011.

[39] N. Swamy, M. Hicks, S. Tse, and S. Zdancewic. Managing policy updates in security-typed languages. In *19th IEEE Computer Security Foundations Workshop (CSFW)*, pages 202–216, July 2006.

[40] G. Takeuti. *Proof Theory*. Dover Books on Mathematics. Dover Books, 1987. Second Edition, republished by Dover Books in 2013. Originally published by North-Holland, Amsterdam.

[41] D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.

[42] L. Waye, P. Buiras, D. King, S. Chong, and A. Russo. It's my privilege: Controlling downgrading in DC-labels. In *Proceedings of the 11th International Workshop on Security and Trust Management*, Sept. 2015.

[43] E. Z. Yang. Logitext, 2012. Accessed February 19, 2019.

[44] S. Zdancewic and A. C. Myers. Robust declassification. In *14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 15–23, June 2001.

[45] L. Zheng and A. C. Myers. End-to-end availability policies and noninterference. In *18th IEEE Computer Security Foundations Workshop (CSFW)*, pages 272–286, June 2005.

A Examples of Permission Models

In Section 4 we saw how FLAFOL can be used to reason about a capabilities-based system. However, FLAFOL's flexibility allows it to model many other kinds of systems. In this appendix, we explore modeling two other systems in FLAFOL: a simple system with no additional assumptions, and a system similar to military classification levels or FLAM's system.

There is no particular reason for there to be some external model of permissions. The "default" permission model simply gives meaning to CanRead and CanWrite through their behavior. That is, the only properties FLAFOL assumes about CanRead and CanWrite are variance constraints, while all other properties of CanRead and CanWrite come from formulae in the context of a proof. This is appropriate in many cases. For instance, in the example of viewing photos on social media, CanRead and CanWrite have their behavior tuned by Bob's selections on his account settings page. It is appropriate for the behaviors based on the selections to be axiomatized directly, rather than forced into some other model. Note that since we only care about confidentiality in that example, CanWrite can have a trivial implementation:

$$p \textbf{ says}_{\ell'} \mathsf{CanWrite}(q, \ell) \leftrightarrow \mathsf{True}.$$

FLAFOL can encode a more-concrete possible permission model by assigning every principal a label representing "which data this person is allowed to read or write." This model appears in the real world in the U.S. military, where every person has a clearance label, and they are allowed to read documents labeled at or

$$s \in \{+, -\} \qquad \overline{+} = - \qquad \overline{-} = +$$

$$\frac{}{\varphi^s \leq \varphi^s} \qquad \frac{\varphi^s \leq \psi^{s'} \qquad \psi^{s'} \leq \chi^{s''}}{\varphi^s \leq \chi^{s''}} \qquad \frac{}{\varphi^s \leq (\varphi \vee \psi)^s} \qquad \frac{}{\psi^s \leq (\varphi \vee \psi)^s} \qquad \frac{}{\varphi^s \leq (\varphi \wedge \psi)^s}$$

$$\frac{}{\psi^s \leq (\varphi \wedge \psi)^s} \qquad \frac{}{\varphi^{\overline{s}} \leq (\varphi \to \psi)^s} \qquad \frac{}{\psi^s \leq (\varphi \to \psi)^s} \qquad \frac{}{(\varphi[x \mapsto t])^- \leq (\forall x\!:\!\sigma.\, \varphi)^-}$$

$$\frac{}{\varphi^+ \leq (\forall x\!:\!\sigma.\, \varphi)^+} \qquad \frac{}{\varphi^- \leq (\exists x\!:\!\sigma.\, \varphi)^-} \qquad \frac{}{(\varphi[x \mapsto t])^+ \leq (\exists x\!:\!\sigma.\, \varphi)^+} \qquad \frac{}{\varphi^s \leq (p \;\textsf{\textbf{says}}_\ell\; \varphi)^s}$$

Figure 8: Signed Subformula Relation

below their clearance. A more subtle version of this model separates reading and writing into confidentiality and integrity labels and allows every principal to have their own idea of each person's label. This is similar to FLAM's model, though our version is typed and does not force principals and labels to be the same.

We can formalize this by giving projection functions from both principals and labels to both confidentiality and integrity. The $\pi_{P,C}$ and $\pi_{P,I}$ projections take principals and produce confidentiality and integrity, respectively, and $\pi_{L,C}$ and $\pi_{L,I}$ do the same, but with labels as arguments. We can think of $\pi_{P,C}(p)$ as "the most confidential data that p can read," while $\pi_{P,I}(p)$ is "the highest integrity data that p can write." We think of $\pi_{L,C}(\ell)$ as "the confidentiality component of label ℓ," while $\pi_{L,I}(\ell)$ is "the integrity component of label ℓ." With these functions, we can say that

$$p \;\textsf{\textbf{says}}_{\ell'}\; \mathsf{CanRead}(q, \ell) \leftrightarrow p \;\textsf{\textbf{says}}_{\ell'}\; (\pi_{L,C}(\ell) \sqsubseteq \pi_{P,C}(q)),$$

and

$$p \;\textsf{\textbf{says}}_{\ell'}\; \mathsf{CanWrite}(q, \ell) \leftrightarrow p \;\textsf{\textbf{says}}_{\ell'}\; (\pi_{P,I}(q) \sqsubseteq \pi_{L,I}(\ell)).$$

The reversal of the order here comes from the fact that integrity, as a flow ordering, is dual to confidentiality.

B Signed Subformulae

As we mention in Section 5.1, FALFOL formulae tend not to move between the left-hand side of the turnstile and the right-hand side. Moreover, the only exceptions to this are the implication rules. This means that looking at where a subformula appears in a sequent tells us which side of the turnstile it can appear on for the rest of the proof. Figure 8 contains the complete rules for the *signed subformula relation* we mention in Section 5.1.

Note that every subformula of a signed formula has a unique sign. If a subformula appears by itself in a sequent during a proof, then which side of the turnstile it is on is determined by its sign. This structure results in the following formal property.

Theorem 6 (Left Signed-Subformula Property)**.** *If $\Gamma \vdash \varphi \mathbin{@} g_1$ appears in a proof of $\Delta \vdash \psi \mathbin{@} g_2$, then for all $\chi_1 \mathbin{@} g_3 \in \Gamma$, either (1) $\chi_1^- \leq \psi^+$ or (2) there is some $\chi_2 \mathbin{@} g_4 \in \Delta$ such that $\chi_1^- \leq \chi_2^-$.*

This proof follows by induction on the FLAFOL proof rules. Details are available in the Coq code.

Many logics also have a similar *right* signed-subformula property. FLAFOL does not enjoy that property since $\Gamma \vdash \varphi \mathbin{@} g_1$ may be a side condition on a forward or a variance rule, and thus not related directly to ψ.

C Simulation

In (multi-)modal logics, we are interested in modeling *perfect* reasoners. That is, reasoners should reason correctly based on their assumed beliefs; if their assumed beliefs were true, then all of their derived beliefs would be as well.

In most logics (which do not have generalized principals, this is axiomatized as a rule in the system, written as follows:

$$\frac{\Gamma \vdash \varphi}{p \textsf{ says}_\ell\ \Gamma \vdash p \textsf{ says}_\ell\ \varphi}$$

Here, $p \textsf{ says}_\ell\ \Gamma$ refers to a copy of Γ with $p \textsf{ says}_\ell$ in front of every formula in Γ. In such logics, this is the main rule for manipulating says statements. However, this requires removing all beliefs that are not those of p at level ℓ in a context before using this rule to reason as p at level ℓ.

FLAFOL instead uses the **says** introduction rules in Section 4, which allows us to retain the beliefs of other principals and of p at other labels, making it easier to discuss communication. This difference causes no harm. FLAFOL reasoners are still be perfect reasoners, which we show by proving a theorem analogous to the above rule. We refer to this as the *simulation* theorem, since it says that p is correctly simulating the world in its head.

Adopting the above rule directly fails for two reasons. The first is that our belief syntax pushes **says** statements into generalized principals, so we must place the new principal-label pair at the beginning of the generalized principal instead of on the formula. The second is that the semantics of implications in FLAFOL mean that $p \textsf{ says}_\ell\ (\varphi \to \psi)$ has different semantics from $(p \textsf{ says}_\ell\ \varphi) \to (p \textsf{ says}_\ell\ \psi)$. To address this concern, we define the $\odot$ operator:

$$p\langle\ell\rangle \odot \varphi \triangleq \begin{cases} (p \textsf{ says}_\ell\ (p\langle\ell\rangle \odot \psi)) \to (p\langle\ell\rangle \odot \chi) & \varphi = \psi \to \chi \\ (p\langle\ell\rangle \odot \psi) \wedge (p\langle\ell\rangle \odot \chi) & \varphi = \psi \wedge \chi \\ (p\langle\ell\rangle \odot \psi) \vee (p\langle\ell\rangle \odot \chi) & \varphi = \psi \vee \chi \\ \forall x : \sigma.\, (p\langle\ell\rangle \odot \psi) & \varphi = \forall x : \sigma.\, \psi \\ \exists x : \sigma.\, (p\langle\ell\rangle \odot \psi) & \varphi = \exists x : \sigma.\, \psi \\ \varphi & \text{otherwise} \end{cases}$$

This essentially "repairs" implications to have the right **says** statements in front of the premise.

We allow the same syntax to prepend to a generalized principal, defining

$$p\langle\ell\rangle \odot (\langle\rangle \cdot g') \triangleq \langle\rangle \cdot p\langle\ell\rangle \cdot g'.$$

We can therefore lift the operator to beliefs straightforwardly, defining $p\langle\ell\rangle \odot (\varphi \mathbin{@} g) \triangleq (p\langle\ell\rangle \odot \varphi) \mathbin{@} p\langle\ell\rangle \odot g$, and from there to contexts as:

$$p\langle\ell\rangle \odot \Gamma \triangleq \begin{cases} \cdot & \Gamma = \cdot \\ (p\langle\ell\rangle \odot \Gamma'), p\langle\ell\rangle \odot (\varphi \mathbin{@} g) & \Gamma = \Gamma', \varphi \mathbin{@} g \end{cases}$$

With this definition in hand, we can now state the simulation theorem in full:

Theorem 7 (Simulation)**.** *The following rule is admissible:*

$$\frac{\Gamma \vdash \varphi \mathbin{@} g}{(p\langle\ell\rangle \odot \Gamma) \vdash p\langle\ell\rangle \odot (\varphi \mathbin{@} g)}$$

The proof is available in the Coq code.

D Details of the Coq proofs

In this appendix, we give some basic guidance to the accompanying Coq code, which can be found at `https://github.com/FLAFOL/flafol-coq`.

General Structure. In the file `Term.v` we define the term language used by FLAFOL along with its type system. In the same file the module GroundInfo is defined. This module takes as parameters information necessary to instantiate the recipe specified in Section 3. For instance, it assumes the existence of a type of sorts and the existence of two sorts: Principal and Label. It also assumes that fresh variables can be generated and that equality of function and relation symbols is decidable. In the file `Formula.v` we define FLAFOL formulae.To simplify proofs, we use a locally nameless representation of variables [13] and binding, and we prove some basic results about this binding discipline. Note also that the definition of FLAFOL formulae is slightly different then that in the paper; rather than being part of the set $\mathcal{R}$, the permission relations are baked into the syntax of FLAFOL formulae directly.

We define the FLAFOL proof system in the file `Sequent.v`. There are three ways in which our Coq formalism differs from the presentation of FLAFOL in Section 4: (1) we use an equivalent presentation of the structural rules, (2) we use a slightly more general logic, and (3) we use two representations of the logic.

First, as is suggested by Pfenning [33], we drop the structural rules from the logic (WEAKENING, EXCHANGE and CONTRACTION), modify our rules so that they never erase anything from the context and we prove that the removed rules are admissible. This makes meta-theoretic proofs simpler.

Second, the logic described in the Coq is slightly more general than the one described in the paper. In the Coq version the ground generalized principal has a label attached to it. Originally we added ground-level labels to accommodate features that we left for future work, but we do not need them for this version of FLAFOL. To show that this is a generalization, for any FLAFOL proof without ground labels, we can simply assign the same ground label to every belief in the proof and acquire a valid proof in the Coq version.

Third, we have two representations of our logic. The first is an (untyped) term language with the appropriate typing rules, and the second is a dependent inductive type. The untyped version eases reasoning about equality, reduces compilation time, and makes proving the admissibility of weakening and substitution easier. The typed version is easier to write automation tactics for. We have proved that both representations are equivalent.

Details of Cut Admissibility Proof. In `NormalForm.v` we define a normal form for FLAFOL proofs. The cut-elimination procedure uses normalization as an essential step.[9] A proof is in normal form if all rules which do not manipulate formulae are higher in the proof tree than those which do. Formally, we define 2 normal forms, first and second normal form, which represent "might use formula-manipulating rules" and "will not use formula manipulating rules", respectively. A proof is in first normal form if, when a rule which manipulates something other than a formula is used, all subproofs above that rule are in second normal form, while a proof is in second normal form it if never uses any rules which manipulate formulae. The main result in this file is that every FLAFOL proof has a normal form.

Theorem 8 (FLAFOL Normal Form)**.** *If $\Gamma \vdash \varphi$ @ g is provable, then it is provable with a proof in normal form.*

Lastly the file `Cut.v` contains the cut-elimination procedure. First we normalize both proofs. If they're both in First Normal Form but not in Second Normal Form, we proceed as Pfenning suggests in [33]: nested triple induction on the formula being cut and on both proofs. If one of them is in Second Normal Form we use a different procedure. This procedure consists of getting the dual rule to the last rule used in the proof that is in Second Normal Form (e.g. VARL for the VARR case) and make it the last rule to the other proof. Due to

[9] In the literature, "normal proof" refers to a cut-free proof, rather than a proof in FLAFOL's normal form.

$$
\textsc{SaysL}\ \dfrac{\textsc{ImpR}'\ \dfrac{\textsc{ImpL}'\ \dfrac{\textsc{SaysL}\ \dfrac{\textsc{Ax}\ \dfrac{}{\varphi \mathbin{@} g \cdot p\langle \ell \rangle \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle}}{p \ \textbf{says}_\ell\ \varphi \mathbin{@} g \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle} \qquad \dfrac{\dfrac{}{\psi \mathbin{@} g \cdot p\langle \ell \rangle \vdash \psi \mathbin{@} g \cdot p\langle \ell \rangle}\ \textsc{Ax}}{\psi \mathbin{@} g \cdot p\langle \ell \rangle \vdash p\ \textbf{says}_\ell\ \psi \mathbin{@} g}\ \textsc{SaysR}}{(\varphi \to \psi) \mathbin{@} g \cdot p\langle \ell \rangle, p\ \textbf{says}_\ell\ \varphi \mathbin{@} g \vdash p\ \textbf{says}_\ell\ \psi \mathbin{@} g}}{(\varphi \to \psi) \mathbin{@} g \cdot p\langle \ell \rangle \vdash (p\ \textbf{says}_\ell\ \varphi) \to (p\ \textbf{says}_\ell\ \psi) \mathbin{@} g}}{p\ \textbf{says}_\ell\ (\varphi \to \psi) \mathbin{@} g \vdash (p\ \textbf{says}_\ell\ \varphi) \to (p\ \textbf{says}_\ell\ \psi) \mathbin{@} g}
$$

Figure 9: Proof that $\textsc{ImpL}'$ and $\textsc{ImpR}'$ allow **says** to distribute over implication.

$$
\textsc{SaysR}\ \dfrac{\textsc{ImpR}'\ \dfrac{\textsc{ImpL}'\ \dfrac{\textsc{SaysR}\ \dfrac{\textsc{Ax}\ \dfrac{}{\varphi \mathbin{@} g \cdot p\langle \ell \rangle \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle}}{\varphi \mathbin{@} g \cdot p\langle \ell \rangle \vdash p\ \textbf{says}_\ell\ \varphi \mathbin{@} g} \qquad \dfrac{\dfrac{}{\psi \mathbin{@} g \cdot p\langle \ell \rangle \vdash \psi \mathbin{@} g \cdot p\langle \ell \rangle}\ \textsc{Ax}}{p\ \textbf{says}_\ell\ \psi \mathbin{@} g \vdash \psi \mathbin{@} g \cdot p\langle \ell \rangle}\ \textsc{SaysL}}{(p\ \textbf{says}_\ell\ \varphi) \to (p\ \textbf{says}_\ell\ \psi) \mathbin{@} g, \varphi \mathbin{@} g \cdot p\langle \ell \rangle \vdash \psi \mathbin{@} g \cdot p\langle \ell \rangle}}{(p\ \textbf{says}_\ell\ \varphi) \to (p\ \textbf{says}_\ell\ \psi) \mathbin{@} g \vdash (\varphi \to \psi) \mathbin{@} g \cdot p\langle \ell \rangle}}{(p\ \textbf{says}_\ell\ \varphi) \to (p\ \textbf{says}_\ell\ \psi) \mathbin{@} g \vdash p\ \textbf{says}_\ell\ (\varphi \to \psi) \mathbin{@} g}
$$

Figure 10: Proof that $\textsc{ImpL}'$ and $\textsc{ImpR}'$ allow **says** to undistribute over implication.

the covariant-contravariant nature of these rules and their duals, this is always possible. For more details see lemmas Cut_h1MCR and Cut_h2MCR in `Cut.v`

Non-Interference. In `Speaksfor.v` we define the relations SF, CanInfl, define the function $\mathcal{G}$ and prove Theorem 4. The compatible supercontexts rules are defined in `CompatibleSuperContext.v`. Finally, the Coq proof of Non-Interference is in `Noninterference.v`; it closely follows the pen-and-paper proof sketched in Section 6.

Simulation. The file `Simulation.v` contains the definition of the function $\odot$ a proof of the Simulation Theorem (Theorem 7).

E Why FLAFOL Does Not Allow Says to Distribute Over Implications

In this appendix, we explore allowing **says** statements to distribute over implications. In particular, we consider replacing the rules ImpL and ImpR with the following rules:

$$
\textsc{ImpL}'\ \frac{\Gamma \vdash \varphi \mathbin{@} g \qquad \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \to \psi) \mathbin{@} g \vdash \chi \mathbin{@} g'} \qquad\qquad \textsc{ImpR}'\ \frac{\Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \to \psi \mathbin{@} g}
$$

Doing so allows us to prove that **says** distributes over implications, as we can see in Figure 9. It also allows ut to prove that **says** *un-distributes* over implication, as we see in Figure 10. While all of ImpL′, ImpR′, and the **says** distribution results may look sensible at first glance, they turn out to cause security bugs and make cut elimination impossible.

To see why, imagine that there are three principals of interest: Alice, Bob, and Cathy, and three labels: ℓ_0, ℓ_1, and ℓ_2, representing Public, Private, and TopSecret, respectively. (We use the shorter names for the sake of readability of formal proofs.) Anybody in the system can read public data (i.e., data labeled with ℓ_0). Alice and Cathy believe all three principals of interest can read private data (i.e., data labeled with ℓ_1), but Bob is unsure of the security clearances and will only send public data to other principals. Alice and Cathy also have top secret clearance, but Bob does not, so he *cannot* read data labeled at ℓ_2. Finally, Bob serves as a redactor: given φ—which represents a document containing private information—he can produce ψ—which represents a redacted version of the same document—performing a declassification in the process.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \varphi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle \vdash \varphi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle}\,\textsc{Ax} \qquad \dfrac{}{\Gamma, \varphi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle, \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle}\,\textsc{Ax}}{\Gamma, (\varphi \to \psi) \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle, \varphi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle}\,\textsc{ImpL}'}{\Gamma, (\varphi \to \psi) \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle, \varphi \mathbin{@} \mathsf{Cathy}\langle \ell_2 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle}\,\textsc{FwdL}^{\dagger}$$

Figure 11: Alice using Cathy's φ and a redaction function

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \varphi \mathbin{@} \mathsf{Bob}\langle \ell_1 \rangle \vdash \varphi \mathbin{@} \mathsf{Bob}\langle \ell_1 \rangle}\,\textsc{Ax}}{\Gamma, \varphi \mathbin{@} \mathsf{Bob}\langle \ell_1 \rangle \vdash \mathsf{Bob}\ \mathbf{says}_{\ell_1}\ \varphi \mathbin{@} \langle\rangle}\,\textsc{SaysR} \qquad \dfrac{\dfrac{}{\Gamma, \psi \mathbin{@} \mathsf{Bob}\langle \ell_0 \rangle \vdash \psi \mathbin{@} \mathsf{Bob}\langle \ell_0 \rangle}\,\textsc{Ax}}{\Gamma, \mathsf{Bob}\ \mathbf{says}_{\ell_0}\ \psi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} \mathsf{Bob}\langle \ell_0 \rangle}\,\textsc{SaysL}}{\Gamma, (\mathsf{Bob}\ \mathbf{says}_{\ell_1}\ \varphi) \to (\mathsf{Bob}\ \mathbf{says}_{\ell_0}\ \psi) \mathbin{@} \langle\rangle, \varphi \mathbin{@} \mathsf{Bob}\langle \ell_1 \rangle \vdash \psi \mathbin{@} \mathsf{Bob}\langle \ell_0 \rangle}\,\textsc{ImpL}'}{\Gamma', \varphi \mathbin{@} \mathsf{Bob}\langle \ell_1 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_0 \rangle}\,\textsc{FwdR}^{\dagger}}{\Gamma', \varphi \mathbin{@} \mathsf{Alice}\langle \ell_1 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_0 \rangle}\,\textsc{FwdL}^{\dagger}}{\Gamma', \varphi \mathbin{@} \mathsf{Alice}\langle \ell_0 \rangle \vdash \psi \mathbin{@} \mathsf{Alice}\langle \ell_0 \rangle}\,\textsc{VarL}^{\dagger}}{\Gamma' \vdash \varphi \to \psi \mathbin{@} \mathsf{Alice}\langle \ell_0 \rangle}\,\textsc{ImpR}'}{\Gamma' \vdash \varphi \to \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle}\,\textsc{VarR}^{\dagger}$$

Figure 12: Proof corresponding to Alice sending φ to Bob and receiving a ψ back

The information needed to formalize these permission policies in our proof are in the context below:

$$\begin{aligned}
\Gamma = {} & \forall p\!:\!\mathsf{Principal}.\, p\ \mathbf{says}_{\ell_1}\ \ell_0 \sqsubseteq \ell_1 \mathbin{@} \langle\rangle, \\
& \forall p\!:\!\mathsf{Principal}.\, p\ \mathbf{says}_{\ell_2}\ \ell_1 \sqsubseteq \ell_2 \mathbin{@} \langle\rangle, \\
& \mathsf{CanRead}(\mathsf{Bob}, \ell_1) \mathbin{@} \mathsf{Alice}\langle \ell_1 \rangle, \\
& \mathsf{CanRead}(\mathsf{Alice}, \ell_2) \mathbin{@} \mathsf{Cathy}\langle \ell_2 \rangle, \\
& \forall p, q\!:\!\mathsf{Principal}.\, p\ \mathbf{says}_{\ell_0}\ \mathsf{CanRead}(q, \ell_0) \mathbin{@} \langle\rangle, \\
& \forall p, q\!:\!\mathsf{Principal}.\, \forall \ell, \ell'\!:\!\mathsf{Label}.\, p\ \mathbf{says}_{\ell}\ \mathsf{CanWrite}(q, \ell') \mathbin{@} \langle\rangle
\end{aligned}$$

To represent Bob's ability to redact information from the document φ, we add one additional belief:

$$\Gamma' = \Gamma, (\mathsf{Bob}\ \mathbf{says}_{\ell_1}\ \varphi) \to (\mathsf{Bob}\ \mathbf{says}_{\ell_0}\ \psi) \mathbin{@} \langle\rangle$$

Imagine further that Alice decides she wants to redact private information from a $\mathsf{TopSecret}$ version of φ that she receives from Cathy, but leave it $\mathsf{TopSecret}$. If she can figure out how to get a redaction implication, she'll simply receive φ from Cathy and then use the implication. This is the proof in Figure 11. Note that, for the sake of brevity and readability, we do not explicitly state side conditions which are proven straightforwardly from Γ. The rules where these side conditions should appear are marked with "†."

While she knows how to *use* and implication representing redaction, Alice does not know how to redact φ except by giving it to Bob. Using $\textsc{ImpL}'$ and $\textsc{ImpR}'$, she is able to package up the process "give Bob a secret version of φ, get back a public version of ψ, and then use variance to get a private version of ψ" as a belief $\varphi \to \psi \mathbin{@} \mathsf{Alice}\langle \ell_1 \rangle$. She can then use variance again to get a belief $\varphi \to \psi \mathbin{@} \mathsf{Alice}\langle \ell_2 \rangle$. This is the proof in Figure 12. Again, we elide side conditions which are proved straightforwardly from Γ, and mark the rules where they should appear with "†."

Cutting these two proofs together gives Alice what she wants: a $\mathsf{TopSecret}$ version of ψ. However, this cut is not possible to eliminate! Examining this through a propositions-as-types lens tells us why: one of Alice or Cathy must send a $\mathsf{TopSecret}$ version of φ to Bob, which neither is unwilling to do.

F The Full FLAFOL Proof System

The full FLAFOL proof system can be found in Figure 13.

G Compatible Supercontexts

Figure 14 contains the full rules for compatible super-contexts.

$$\textsc{Ax}\ \frac{}{\Gamma, \varphi \mathbin{@} g \vdash \varphi \mathbin{@} g} \qquad \textsc{Weakening}\ \frac{\Gamma \vdash \psi \mathbin{@} g}{\Gamma, \varphi \mathbin{@} g' \vdash \psi \mathbin{@} g}$$

$$\textsc{Contraction}\ \frac{\Gamma, (\varphi \mathbin{@} g), (\varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'}{\Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'} \qquad \textsc{Exchange}\ \frac{\Gamma, (\varphi \mathbin{@} g_1), (\psi \mathbin{@} g_2), \Gamma' \vdash \chi \mathbin{@} g}{\Gamma, (\psi \mathbin{@} g_2), (\varphi \mathbin{@} g_1), \Gamma' \vdash \chi \mathbin{@} g}$$

$$\textsc{FalseL}\ \frac{}{\Gamma, \mathsf{False} \mathbin{@} g \vdash \varphi \mathbin{@} g \cdot g'} \qquad \textsc{TrueR}\ \frac{}{\Gamma \vdash \mathsf{True} \mathbin{@} g}$$

$$\textsc{AndL}\ \frac{\Gamma, (\varphi \mathbin{@} g), (\psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \wedge \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{AndR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \qquad \Gamma \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \wedge \psi \mathbin{@} g}$$

$$\textsc{OrL}\ \frac{\Gamma, \varphi \mathbin{@} g \vdash \chi \mathbin{@} g' \qquad \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{OrR1}\ \frac{\Gamma \vdash \varphi \mathbin{@} g}{\Gamma \vdash \varphi \vee \psi \mathbin{@} g} \qquad \textsc{OrR2}\ \frac{\Gamma \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \vee \psi \mathbin{@} g}$$

$$\textsc{ImpL}\ \frac{\Gamma \vdash \varphi \mathbin{@} \langle\rangle \qquad \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Gamma, (\varphi \to \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{ImpR}\ \frac{\Gamma, \varphi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \to \psi \mathbin{@} g}$$

$$\textsc{ForallL}\ \frac{\Gamma, \varphi[x \mapsto t] \mathbin{@} g \vdash \psi \mathbin{@} g'}{\Gamma, (\forall x{:}\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'} \qquad \textsc{ForallR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \qquad x \notin \mathsf{FV}(\Gamma, g)}{\Gamma \vdash \forall x{:}\sigma.\, \varphi \mathbin{@} g}$$

$$\textsc{ExistsL}\ \frac{\Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g' \qquad x \notin \mathsf{FV}(\Gamma, \psi, g, g')}{\Gamma, (\exists x{:}\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'} \qquad \textsc{ExistsR}\ \frac{\Gamma \vdash \varphi[x \mapsto t] \mathbin{@} g}{\Gamma \vdash \exists x{:}\sigma.\, \psi \mathbin{@} g}$$

$$\textsc{SaysL}\ \frac{\Gamma, \varphi \mathbin{@} g \cdot p\langle\ell\rangle \vdash \psi \mathbin{@} g'}{\Gamma, p\ \mathbf{says}_\ell\ \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'} \qquad \textsc{SaysR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle}{\Gamma \vdash p\ \mathbf{says}_\ell\ \varphi \mathbin{@} g}$$

$$\textsc{SelfL}\ \frac{\Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''}{\Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''} \qquad \textsc{SelfR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g'}{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot p\langle\ell\rangle \cdot g'}$$

$$\textsc{VarL}\ \frac{\begin{array}{c}\Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell'\rangle \cdot g') \vdash \psi \mathbin{@} g'' \\ \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \ell \sqsubseteq \ell' \mathbin{@} g \cdot p\langle\ell'\rangle\end{array}}{\Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''} \qquad \textsc{VarR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell'\rangle \cdot g' \qquad \Gamma \vdash \ell' \sqsubseteq \ell \mathbin{@} g \cdot p\langle\ell\rangle}{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g'}$$

$$\textsc{FwdL}\ \frac{\begin{array}{c}\Gamma, (\varphi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \chi \mathbin{@} g'' \\ \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{CanRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{CanWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Gamma, \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \vdash \chi \mathbin{@} g''} \qquad \textsc{FwdR}\ \frac{\begin{array}{c}\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \\ \Gamma \vdash \mathsf{CanRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Gamma \vdash \mathsf{CanWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Gamma \vdash \varphi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g'}$$

$$\textsc{FlowsToRefl}\ \frac{}{\Gamma \vdash \ell \sqsubseteq \ell \mathbin{@} g} \qquad \textsc{FlowsToTrans}\ \frac{\Gamma \vdash \ell_1 \sqsubseteq \ell_2 \mathbin{@} g \qquad \Gamma \vdash \ell_2 \sqsubseteq \ell_3 \mathbin{@} g}{\Gamma \vdash \ell_1 \sqsubseteq \ell_3 \mathbin{@} g}$$

$$\textsc{CRVar}\ \frac{\Gamma \vdash \mathsf{CanRead}(p, \ell_2) \mathbin{@} g \qquad \Gamma \vdash \ell_1 \sqsubseteq \ell_2 \mathbin{@} g}{\Gamma \vdash \mathsf{CanRead}(p, \ell_1) \mathbin{@} g}$$

$$\textsc{CWVar}\ \frac{\Gamma \vdash \mathsf{CanWrite}(p, \ell_2) \mathbin{@} g \qquad \Gamma \vdash \ell_2 \sqsubseteq \ell_1 \mathbin{@} g}{\Gamma \vdash \mathsf{CanWrite}(p, \ell_1) \mathbin{@} g}$$

Figure 13: Full FLAFOL Proof System

$$\textsc{CSCRefl}\ \frac{}{\Gamma \ll \Gamma \vdash \varphi \mathbin{@} g} \qquad \textsc{CSCUnion}\ \frac{\Delta_1 \ll \Gamma \vdash \varphi \mathbin{@} g \qquad \Delta_2 \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta_1 \cup \Delta_2 \ll \Gamma \vdash \varphi \mathbin{@} g}$$

$$\textsc{CSCContraction}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g), (\varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'}{\Delta \ll \Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'}$$

$$\textsc{CSCExchange}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g_1), (\psi \mathbin{@} g_2), \Gamma' \vdash \chi \mathbin{@} g}{\Delta \ll \Gamma, (\psi \mathbin{@} g_2), (\varphi \mathbin{@} g_1), \Gamma' \vdash \chi \mathbin{@} g} \qquad \textsc{CSCAndL}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g), (\psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \wedge \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}$$

$$\textsc{CSCAndR1}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \wedge \psi \mathbin{@} g} \qquad \textsc{CSCAndR2}\ \frac{\Delta \ll \Gamma \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \wedge \psi \mathbin{@} g}$$

$$\textsc{CSCOrL1}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{CSCOrL2}\ \frac{\Delta \ll \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}$$

$$\textsc{CSCOrR1}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \vee \psi \mathbin{@} g} \qquad \textsc{CSCOrR2}\ \frac{\Delta \ll \Gamma \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \vee \psi \mathbin{@} g}$$

$$\textsc{CSCImpL1}\ \frac{\Delta \ll \Gamma, \psi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \rightarrow \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'} \qquad \textsc{CSCImpL2}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} \langle\rangle}{\Delta \ll \Gamma, (\varphi \rightarrow \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}$$

$$\textsc{CSCImpR}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \rightarrow \psi \mathbin{@} g} \qquad \textsc{CSCForallL}\ \frac{\Delta \ll \Gamma, \varphi[x \mapsto t] \mathbin{@} g \vdash \psi \mathbin{@} g'}{\Delta \ll \Gamma, (\forall x\!:\!\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'}$$

$$\textsc{CSCForallR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \qquad x \notin \mathsf{FV}(\Gamma, g)}{\Delta \ll \Gamma \vdash \forall x\!:\!\sigma.\, \varphi \mathbin{@} g}$$

$$\textsc{CSCExistsL}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g' \qquad x \notin \mathsf{FV}(\Gamma, \psi, g, g')}{\Delta \ll \Gamma, (\exists x\!:\!\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'} \qquad \textsc{CSCExistsR}\ \frac{\Delta \ll \Gamma \vdash \varphi[x \mapsto t] \mathbin{@} g}{\Delta \ll \Gamma \vdash \exists x\!:\!\sigma.\, \varphi \mathbin{@} g}$$

$$\textsc{CSCSaysL}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} g \cdot p\langle\ell\rangle \vdash \psi \mathbin{@} g'}{\Delta \ll \Gamma, p \ \textsf{\textbf{says}}_\ell\ \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'} \qquad \textsc{CSCSaysR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle}{\Delta \ll \Gamma \vdash p \ \textsf{\textbf{says}}_\ell\ \varphi \mathbin{@} g}$$

$$\textsc{CSCSelfL}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''}{\Delta \ll \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''} \qquad \textsc{CSCSelfR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g'}{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot p\langle\ell\rangle \cdot g'}$$

$$\textsc{CSCVarL}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell'\rangle \cdot g') \vdash \psi \mathbin{@} g'' \qquad \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \ell \sqsubseteq \ell' \mathbin{@} g \cdot p\langle\ell'\rangle}{\Delta \ll \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g''}$$

$$\textsc{CSCVarR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell'\rangle \cdot g' \qquad \Gamma \vdash \ell' \sqsubseteq \ell \mathbin{@} g \cdot p\langle\ell\rangle}{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g'}$$

$$\textsc{CSCFwdL}\ \frac{\begin{array}{c}\Delta \ll \Gamma, (\varphi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \chi \mathbin{@} g'' \qquad \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{CanRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Gamma, (\varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{CanWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Delta \ll \Gamma, \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \vdash \chi \mathbin{@} g''}$$

$$\textsc{CSCFwdR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \qquad \Gamma \vdash \mathsf{CanRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \qquad \Gamma \vdash \mathsf{CanWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle}{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g'}$$

Figure 14: Compatible Supercontext Rules