First-Order Logic for Flow-Limited Authorization

***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 verify them in Coq, 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.**

***Index Terms*—authorization, information flow, logic, proof theory, authorization logic**

I. Introduction

Distributed systems often make authorization decisions based on private data. A public decision might therefore leak private information. Preventing such leakage requires nontrivial reasoning about the interaction between information flow and authorization policies [1]–[3]. 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.

Reasoning about the interaction between authorization and information-flow policies is difficult, partially because they use different notions of trust. Information-flow systems tend to focus on tracking data dependencies by representing an 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., [4]–[6]) 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.

The purpose of both forms of trust is to selectively constrain communication, which 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. Therefore, describing the information security of such authorization decisions requires a nuanced form of trust that includes information-flow policies.

Information flow systems make it easy to track when and what information is communicated from one principal to another. Each transfer of data from one label to another represents a communication. It is less clear in authorization logics when such communications occur. One interpretation might be that if Alice delegates trust to Bob, then she immediately imports all of Bob's beliefs. In fact, several logics (e.g., [6]–[8]) make this explicit by interpreting Bob's beliefs as if they also came from Alice.

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

In addition to being able to express trust and communication, authorization policies are often difficult to express formally. Any tool that combines authorization and information flow should therefore be capable of expressing real-world policies. Nexus [6], [9]—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, to evaluate any attempt to combine authorization with information flow policies, we must examine the resulting security guarantees. Both authorization logics and information-flow systems have security properties called *non-interference*. In information-flow systems this property is standard, while authorization logics often view it as highly desirable but unobtainable. Although the two formulations look quite different, both make guarantees limiting how one component of a system can influence 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 there is a flow relationship between the label of x and the label of y.

Both of these notions of non-interference are important. Consider again the example where Alice attempts to view Bob's photo, but Bob's friend list is private. Bob adding or removing Cathy from his friend list should not affect Alice's

[1]The Nexus Authorization Logic is actually a monadic second-order logic, but this is used only to encode $\textsf{says}$; only first-order quantification is used in any of their examples [6].

beliefs. To enforce this, whether or not Cathy is Bob's friend must not affect the set of Bob's beliefs that Alice *may* learn.

In order to glue together both ideas of non-interference, we must understand the connection between their notions of trust. As we have discussed, these trust notions are difficult to connect, 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,
- has clear 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.

We additionally aim to clarify the foundations of flow-limited authorization. We therefore strive to keep FLAFOL's model of principals, labels, and communication as simple as possible. As a result, we deviate from previous work by not requiring that labels form a lattice (see Sections III and V-A).

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 [2], SecPAL$^+$ [1], [5], and AURA [10], [11]. FLAM models trust using information flow, AURA uses DCC [8], [12], a propositional authorization logic, and SecPAL$^+$ places disconnected information flow labels on principal-based trust policies. Neither FLAM nor SecPAL$^+$ can reason about nested beliefs, and both are severely 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 II we discuss three running examples. This also serves as an intuitive introduction to FLAFOL. In Section III we discuss the system model of FLAFOL, including our minimalist model of information-flow labels. In Section IV we discuss the FLAFOL proof rules in detail. In Section V we discuss the proof theory of FLAFOL, proving several important theorems, including consistency and cut elimination. These theorems are proven in Coq. In Section VI we provide a non-interference theorem for FLAFOL. In Section VII we discuss future work, in Section VIII we discuss related work, and finally in Section IX we conclude.

II. 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. For instance, each defines the meaning of a label 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. 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 III.

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. This can be useful to model situations where each principal has their own view of some piece of data.

A. Viewing Pictures on Social Media

We begin by reconsidering in more detail the example from Section I where Alice requests to view Bob's picture on a social media service. This social media service allows Bob to set privacy policies, and Bob has chosen to make his pictures visible to only his friends. When Alice makes her request, the service can scan Bob's friend list and determine if she is allowed to view the photo. If she is on Bob's friend list and the photo is available, it shows her the photo. If the photo is unavailable, it shows her an HTTP 404: Not Found page. (Of course, since databases are not entirely reliable, it shows her a 404 page sometimes even when the photo is available.) Finally, 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 $\mathsf{Bob\ says\ isFriend(Alice)}$. Since says statements can encompass any formula, we can express the fact that Bob believes that Alice is *not* his friend as $\mathsf{Bob\ says\ \neg isFriend(Alice)}$.

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 IV), 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* believes False will affirm every statement.

Now imagine that, as in Section I, 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 on Bob's friend list, 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. Thus, whether she is not on Bob's friend list, the photo does not exist, or the database does not respond, the social media service must show Alice an HTTP 404 page.

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 that policy to his belief about Alice being his friend, we would write $\mathsf{Bob\ says}_{\mathsf{Friends}}\ \mathsf{isFriend}(\mathsf{Alice})$. Over the next two sections (Sections III and IV) we will develop the technology required for Bob to express that policy.

Notably, including these labels only provides the above semantics because FLAFOL is intuitionistic. In a classical system, we could use the law of the excluded middle to prove at any label—including a public one—that either Alice can see Bob's picture or she cannot. If we encode the result Alice sees using implications (e.g., $\mathsf{Bob\ says}_{\mathsf{Friends}}\ \mathsf{isFriend}(\mathsf{Alice}) \rightarrow \mathsf{show}(\mathsf{pic}, \mathsf{Alice})$), we could then derive that either Alice sees the picture or Alice sees an HTTP 403: Forbidden. Since both results improperly leak information, we should display neither, but in a classical system we would provably show one or the other. This leads us to reject the law of the excluded middle.

B. Preventing SQL Injections with Integrity Tracking

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 knows how to sanitize inputs using a $\mathsf{sanitize}$ function, neutralizing any attacks, so when it encounters a low-integrity input, it is willing to sanitize that input and endorse the result.

While FLAFOL does not support this sort of endorsement directly, its support for arbitrary implications means that we can easily encode it. 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 $\mathsf{System\ says}_{\mathsf{LInt}}\ \mathsf{DBInput}(x)$. Here LInt represent low-integrity beliefs. Now, to represent the system's willingness to endorse any sanitized input, we say

$$\begin{aligned}&\mathsf{System\ says}_{\mathsf{LInt}}\ \mathsf{DBInput}(x) \rightarrow\\ &\qquad \mathsf{System\ says}_{\mathsf{HInt}}\ \mathsf{DBInput}(\mathsf{sanitize}(x))\end{aligned}$$

This type of endorsement also has interesting security ramifications, which we investigate in Section VI.

C. Providing a Hospital Bill in the Presence of Reinsurance

Imagine now that Alice finds herself in the hospital. Luckily she has insurance provided by employer, but her employer just switched insurance companies. She was issued a new insurance card, which she immediately put in her purse. Now she has two unexpired insurance cards, and she can't remember which one is valid. Thus, there are two insurers, I_1 and I_2, either of which may be Alice's insurer.

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, as an insurance expert, also 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 foot 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 $\mathsf{Bob\ says}_{\ell_H}\ (\mathsf{canWrite}(I_1, \ell_H) \vee \mathsf{canWrite}(I_2, \ell_H))$. 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 $\mathsf{Bob\ says}_{\ell_H}\ (I_2\ \mathsf{says}_{\ell_H}\ \mathsf{canWrite}(I_1, \ell_H))$.

Bob's beliefs allow him to prove that I_1 may help determine the contents of Alice's bill; assuming the previous two statements we can prove that $\mathsf{Bob\ says}_{\ell_H}\ \mathsf{canWrite}(I_1, \ell_H)$. 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 $\mathsf{Bob\ says}_{\ell_H}\ \mathsf{canWrite}(I_1, \ell_H)$.

We think of Bob as performing this proof, since it is a proof that entirely is 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, one of the most important constructs in the formal presentation of FLAFOL (see Section III).

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 much more difficult, as we will see in Section VI.

D. 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 [13] describes

$$
\begin{array}{crcl}
\text{Sorts} & \sigma & ::= & \mathsf{label} \mid \mathsf{principal} \mid \cdots \\
\text{Labels} & \ell & & \\
\text{Principals} & p, q, r & & \\
\sigma\text{-terms} & t & ::= & x \mid f(t_1, \ldots, t_n) \\
\text{Formulae} & \varphi, \psi, \chi & ::= & R(t_1, \ldots, t_n) \\
& & \mid & \mathsf{True} \mid \mathsf{False} \\
& & \mid & \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \\
& & \mid & \forall x{:}\sigma.\, \varphi \mid \exists x{:}\sigma.\, \varphi \\
& & \mid & p \;\mathsf{says}_\ell\; \varphi \\
& & \mid & \ell_1 \sqsubseteq \ell_2 \\
& & \mid & \mathsf{canRead}(p, \ell) \\
& & \mid & \mathsf{canWrite}(p, \ell) \\
\begin{array}{c}\text{Generalized}\\ \text{Principals}\end{array} & g & ::= & \langle\rangle \mid g \cdot p\langle \ell \rangle
\end{array}
$$

Fig. 1. FLAFOL Syntax

how to use information-flow labels to create safe transactions across databases with different consistency models, and the work of Zheng and Myers [14] 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.

In our last example we used the relation canWrite to determine who may affect the contents of Alice's hospital bill, and in our first example we could have expressed our confidentiality permissions using a similar canRead relation. These relations form FLAFOL's (very abstract) notion of trust. By default, canRead and canWrite gain meaning only through their behavior in a context. As we discuss in Appendix E, they can also encode capabilities and FLAM's model of trust.

III. System Model

In FLAFOL, terms are divided into different types, called *sorts* in the tradition of logic. Formally, FLAFOL is parameterized on a set of sorts which must contain at least principal and label. It is also parameterized on a set of function symbols $\mathcal{F}$ and a set of relation symbols $\mathcal{R}$.

Terms t in FLAFOL are either variables or function applications, which consist of a *function symbol* $f \in \mathcal{F}$ and zero or more arguments. We encode constants as functions with no arguments. For instance, the principal constant Alice is formally a nullary function into principal. Both logic and functional programming commonly view constants this way.

Atomic formulae in FLAFOL are either True, False, or a relation, which consists of a *relation symbol* $R \in \mathcal{R}$ and zero or more parameters, each of which is a FLAFOL term. We have three required relations that we discuss below: flows-to ($\sqsubseteq$), canRead, and canWrite. Figure 1 contains the complete syntax of FLAFOL formulae. For brevity we assume that all FLAFOL terms and formulae are well-sorted.

We assume no particular function symbols for either principal or label, but as mentioned, we do assume three relations on principals and labels. Flows-to relates two labels, while canRead and canWrite relate a principal and a label.

$$\textsc{FlowsToRefl}\ \frac{}{\Gamma \vdash \ell \sqsubseteq \ell \mathbin{@} g}$$

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

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

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

Fig. 2. Flows-To and Permission Rules

We refer to these three relations as *permissions* because they define the trust relationships governing communication between principals.

The flows-to relation is reflexive and transitive, making the label sort a preorder. Intuitively, if $\ell_1 \sqsubseteq \ell_2$ then data labeled ℓ_1 can affect data labeled ℓ_2. If Alice can read a piece of data A with label ℓ_2, she may learn information about data with label ℓ_1 used to calculate A. This means she should 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.

Existing information-flow tools often 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. Since all lattices form preorders, our results are entirely compatible with lattice-based label models. Indeed, in Section V-A we show that enforcing a lattice structure on FLAFOL's labels is both simple and logically consistent.

This effort to minimize the system constraints extends to other areas as well. As we noted in Section II, the fact that permission relations are governed primarily by assumptions placed in the context Γ allows FLAFOL to reason about systems with complex and varied permission models. Labels may represent any combination of different security policies (e.g., confidentiality, integrity, availability, etc.) and, critically, principals may disagree with each other, including on the permission relations. Additionally, FALFOL's ability to handle arbitrary sorts—like integers or capability tokens—and function and relation symbols allow it to straightforwardly model numerous system components. For instance, in Section II-B we used the unary relation DBInput on sort data to represent that a piece of data is an input, and the sanitize : data $\rightarrow$ data function symbol to represent a sanitization operation.

Figure 2 shows the rules that enforce the flows-to preorder and the variance for canRead and canWrite. 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 II-C 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. 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*. Figure 1 contains the formal grammar for generalized principals.

Every formula φ in a FLAFOL proof is paired with a generalized principal g who believes the formula, written φ @ g. This gives us the ability to write rules that work for all possible reasoners.

Applying FLAFOL. FLAFOL can help ensure that a system's authorization mechanism does not leak information. If we represent the components of the system as principals and all information as beliefs using appropriate sorts, relations, and function symbols, we can encode an authorization request as a relation. For example, in Section II-A, we encode Bob's friend list and his privacy policies as a set of beliefs—for instance, $\mathsf{isFriend}(\mathsf{Alice})$ @ $\mathsf{Bob}\langle\mathsf{Friends}\rangle$ records that Alice is on Bob's friend list, and that Bob's friend list is only visible to his friends. We further encode permission to view a picture using a binary relation $\mathsf{canView}$ that relates a $\mathsf{picture}$ sort and principals. A proof that $\mathsf{canView}(x, \mathsf{Alice})$ would then indicate that Alice is authorized to view picture x.

With such an encoding, a proof that any request authorized by the system has a valid FLAFOL proof would demonstrate that the system can validate the authorization without leaking information. One way to make such an assurance would be for the system to use proof-carrying authorization. That is, any request must contain a FLAFOL proof that the request is authorized. Because FLAFOL is an intuitionistic sequent calculus, such a system could use an off-the-shelf proof search algorithm, such as Andreoli's Focusing proof search [15].

IV. 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 A. 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 III. Readers familiar with sequent 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:

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

$$\textsc{AndR}\ \frac{\Gamma \vdash \varphi \mathbin{@} g \qquad \Gamma \vdash \psi \mathbin{@} g}{\Gamma \vdash \varphi \land \psi \mathbin{@} g}$$

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 $\varphi \land \psi$ @ g by splitting it into two assumptions, one for each conjunct, while the AndR rule takes proofs of two formulae and proves their conjunction.[2]

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 $p\ \mathsf{says}_\ell\ (\varphi \lor \psi)$, we can prove $(p\ \mathsf{says}_\ell\ \varphi) \lor (p\ \mathsf{says}_\ell\ \psi)$. In an intuitionistic logic like FLAFOL,[3] disjunctions must be a proof of one side or the other. The proof of distribution of says over or 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.

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 $p\ \mathsf{says}_\ell\ (b = 0 \lor b = 1)$, we should not be able to show that $(p\ \mathsf{says}_\ell\ b = 0) \lor (p\ \mathsf{says}_\ell\ b = 1)$. A NuPRL-like "squash" operator, which prevents evidence from being used [17], could model this, but further research is needed for FLAFOL to reason about the security of such protocols.

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., $p\ \mathsf{says}_\ell\ (\varphi \to \psi)$

[2]For readers interested in learning more about sequent calculus, we recommend MIT's interactive tool for teaching sequent caluclus as a tutorial [16].

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

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

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

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

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

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

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

Fig. 3. Selected FLAFOL Proof Rules

does not imply that $(p \mathsf{\ says}_\ell\ \varphi) \to (p \mathsf{\ says}_\ell\ \psi)$. Instead, $p \mathsf{\ says}_\ell\ (\varphi \to \psi)$ implies $\varphi \to (p \mathsf{\ says}_\ell\ \psi)$. We can therefore think of implications as *tests* of the system state. That is, if a generalized principal g believes $\varphi \to \psi$, g can run a test that allows it to observe ψ whenever φ is true about the system.

We can still form implications about generalized principal's beliefs, but we must insert appropriate says statements into the premise to do so. In Section V-D, we discuss how this implication 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, and each has a corresponding left rule, which acts contravariantly and can be found in Appendix A.

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 [18], [19], but it remains an unusual choice.

Authorization-logic-style communication is provided by the forwarding rule FwdR. 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 $p \mathsf{\ says}_\ell\ \mathsf{canRead}(q, \ell)$, and
- q is willing to allow p to determine its beliefs at label ℓ, denoted $q \mathsf{\ says}_\ell\ \mathsf{canWrite}(p, \ell)$.

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

V. 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. We also develop a new proof-theoretic tool, *compatible supercontexts*, which we use in our non-interference theorem in Section VI and discuss fundamental limitations that inform our unusual implication semantics. Importantly, every theorem in this section is verified in the Coq proof assistant [20].

A. 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 a lot of useful structure to proofs. To handle implications, however, we must keep track of their nesting structure, which we do by considering *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.

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

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

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

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

Fig. 4. Selected rules for the Signed Subformula Relation

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. Perhaps confusingly, 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*

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

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

The proof follows by a simple induction on the FLAFOL proof rules. Details are available in the Coq code.

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

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

Proof. If Γ is empty, then the "for all" 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., $\forall \ell \colon \mathsf{label}.\, \ell \sqsubseteq \top$) 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.

B. Compatible Supercontexts

To prove Theorem 1 we needed to consider the possible locations of *formulae* within a sequent, but in Section VI 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 C.

$$\text{CSCRefl}\ \frac{}{\Gamma \ll \Gamma \vdash \varphi \mathbin{@} g}$$

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

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

$$\text{CSCImpR}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} \langle\rangle \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \rightarrow \psi \mathbin{@} g}$$

Fig. 5. Selected Rules for Compatible Supercontexts

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 $\Delta \ll \Gamma \vdash \varphi \mathbin{@} g$.*

C. 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. Not only does it allow us to create lemmata to be used in a proof, it allows us to prove some things that do not obviously have other proofs. For instance, consider the following rule.

$$\text{UnsaysR}\ \frac{\Gamma \vdash p \ \mathsf{says}_{\ell}\ \varphi \mathbin{@} g}{\Gamma \vdash \varphi \mathbin{@} g \cdot p\langle \ell \rangle}$$

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 $\Gamma \vdash p \text{ says}_\ell\ \varphi\ @\ g$ with the following proof:[4]

$$\dfrac{\dfrac{}{\varphi\ @\ g \cdot p\langle\ell\rangle \vdash \varphi\ @\ g \cdot p\langle\ell\rangle}\ \textsc{Ax}}{p \text{ says}_\ell\ \varphi\ @\ g \vdash \varphi\ @\ g \cdot p\langle\ell\rangle}\ \textsc{SaysL}$$

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 highly nontrivial. Appendix D contains more details. This theorem is proven in Coq.

This theorem is one of the key theorems of proof theory [21], [22]. Frank Pfenning has called it "[t]he central property of sequent calculi" [23]. From the propositions-as-types perspective, cut elimination is preservation of types under substitution.

D. Implications and Communication

Recall from Section III how we interpret a formula such as $\text{Alice says}_{\text{Public}}\ (\varphi \to \psi)$: if φ is true about the system, then Alice gets to observe ψ. We now have the tools to understand why we use this interpretation. Imagine that we replace rules IMPL and IMPR with rules IMPL$'$ and IMPR$'$ which interpret the above formula as: if Alice believes φ, then Alice also believes ψ. These rules would replace $\varphi\ @\ \langle\rangle$ in IMPL/IMPR with $\varphi\ @\ g$. They would allow us to prove that if $\text{Alice says}_{\text{Public}}\ (\varphi \to \psi)$ then $(\text{Alice says}_{\text{Public}}\ \varphi) \to (\text{Alice says}_{\text{Public}}\ \psi)$.

In this system, imagine that we had a proof of $\Gamma \vdash \text{Alice says}_{\text{Public}}\ (\varphi \to \psi)$ which required Alice to forward φ to Bob, which she is willing to do because she believes φ publicly. Alice is also willing to relabel this belief as private, since this is a public belief. Distributing the says gives a proof of the form $\Gamma \vdash (\text{Alice says}_{\text{Private}}\ \varphi) \to (\text{Alice says}_{\text{Private}}\ \psi)$. If we cut this proof with a proof of $\Gamma \vdash \text{Alice says}_{\text{Private}}\ \varphi$ and then eliminate the cut, we discover that Alice needs to send φ to Bob. Alice, however, may be unwilling to do this since she may only believe φ privately. Because of how FLAFOL interprets $\text{Alice says}_{\text{Public}}\ (\varphi \to \psi)$, when Alice relabels her implication to secret, she is *only relabeling the output*. Thus, any communications that happen in the proof of the implication are still allowed.

We can alternatively adopt a propositions-as-types viewpoint. 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 proof rules suggested in the preceding paragraph would then force functions to behave *covariantly* on their inputs where they should behave *contravariantly*. This 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.

This limitation is fundamental. Given a set of modalities $M_1, M_2, \ldots$ with a preorder $\leq$, it is impossible to have all of:

1) Distributing modalities over implication: for any modality M and formulae φ and ψ, $M(\varphi \to \psi) \vdash (M\varphi) \to (M\psi)$
2) Varying modalities according to the preorder: for any $M_1 \leq M_2$ and φ, $M_1\varphi \vdash M_2\varphi$
3) Proven implications where the proof of the implication uses (2), (variance and forward in FLAFOL).

The combination of (1) and (2) forces implications to treat their premises covariantly, but (3) requires implications to treat their premises contravariantly. A standard type-theoretic argument suggests that this is incoherent. Because (2) is fundamental to FLAFOL's approach to information flow and communication and (3) is important for modeling real-world systems (see Section II-B), we choose to keep (2) and (3) in FLAFOL, leading to our current form of implication.

VI. NON-INTERFERENCE

Both authorization logics and information flow systems have important security properties called *non-interference* [24]–[26]. 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 from one label to another based on the flow relationship between those labels. 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 [8], [26]. No similar theorem reasons about information flow or applies to policies combining discoverable trust and logical disjunction. Theorem 5 does both.

A. Trust in FLAFOL

Building a notion of trust on generalized principals requires us to consider both the trust of the underlying (regular)

[4]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.

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

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

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

$$\textsc{VarSF}\ \frac{\Gamma \vdash \ell \sqsubseteq \ell' \mathbin{@} g \cdot p\langle \ell' \rangle}{\Gamma \vdash g \cdot p\langle \ell \rangle \mathsf{\ sf\ } g \cdot p\langle \ell' \rangle}$$

$$\textsc{FwdSF}\ \frac{\begin{array}{c}\Gamma \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle \ell \rangle \\ \Gamma \vdash \mathsf{canWrite}(p, \ell) \mathbin{@} g \cdot q\langle \ell \rangle\end{array}}{\Gamma \vdash g \cdot p\langle \ell \rangle \mathsf{\ sf\ } g \cdot q\langle \ell \rangle}$$

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

Fig. 6. The rules defining speaks for.

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:*

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

This notion of trust allows us to begin structuring a non-interference statement. Ideally, we would 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 $\Gamma, (\varphi \mathbin{@} g_1) \vdash \psi \mathbin{@} g_2$ is provable, then either $\Gamma \vdash \psi \mathbin{@} g_2$ 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.

B. 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 $p \mathsf{\ says}_\ell\ \varphi \mathbin{@} \langle\rangle \vdash \varphi \mathbin{@} \langle\rangle \cdot p\langle \ell \rangle$, yet we cannot prove $\langle\rangle \mathsf{\ sf\ } \langle\rangle \cdot p\langle \ell \rangle$.

To address this case, we can view $p \mathsf{\ says}_\ell\ \varphi \mathbin{@} \langle\rangle$ 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. This strategy works for simple formulae, but 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 those different generalized principals. We eliminate this concern by considering a *set* of generalized principals referenced in a given belief. We build this set using an operator we denote $\mathcal{G}$.

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

Implications consider only 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 $\Gamma, (\varphi \mathbin{@} g_1) \vdash \psi \mathbin{@} g_2$, then either $\Gamma \vdash \psi \mathbin{@} g_2$, or there is some $g_1' \in \mathcal{G}(\varphi \mathbin{@} g_1)$, $g_2' \in \mathcal{G}(\psi \mathbin{@} g_2)$, and some g_1'' such that $\Gamma \vdash g_1' \cdot g_1'' \mathsf{\ sf\ } g_2'$.

The g_1'' here is needed because the side conditions of the forward and variance rules are statements about prefixes of the final generalized principal.

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 metal-level version of says, meaning similar conversions do not even make sense.

C. 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 II-B.

$$\textsc{SF-CI}\ \frac{\Gamma \vdash g_1\ \mathsf{sf}\ g_2}{\Gamma \vdash g_1\ \mathsf{canInfl}\ g_2} \qquad \textsc{ExtCI}\ \frac{\Gamma \vdash g_1\ \mathsf{canInfl}\ g_2}{\Gamma \vdash g_1 \cdot g'\ \mathsf{canInfl}\ g_2 \cdot g'}$$

$$\textsc{TransCI}\ \frac{\Gamma \vdash g_1\ \mathsf{canInfl}\ g_2 \qquad \Gamma \vdash g_2\ \mathsf{canInfl}\ g_3}{\Gamma \vdash g_1\ \mathsf{canInfl}\ g_3}$$

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

Fig. 7. The rules defining the *can influence* relation.

Recall from Section II-B 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.

$$\begin{aligned}&\mathsf{System\ says_{LInt}\ DBInput}(x) \to \\ &\qquad \mathsf{System\ says_{HInt}\ DBInput}(\mathsf{sanitize}(x))\end{aligned}$$

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 [8] or by explicitly using assumed implications to represent trust [26]. We keep closer to the latter model and make the implicit trust of implications explicit in our statement of non-interference. To do so, we do not use the speaks-for relation, but instead 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 test 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 by 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 [27]–[35]. We believe it would be interesting future work to apply these analyses and restrictions to FLAFOL to produce more precise statements of security.

D. 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 these two features can have in combination, recall from the reinsurance example of Section II-C that Bob believes $\mathsf{canWrite}(I_1, \ell_H)$ if he believes that $\mathsf{canWrite}(I_1, \ell_H) \vee \mathsf{canWrite}(I_2, \ell_H)$ and $I_2\ \mathsf{says}_{\ell_H}\ \mathsf{canWrite}(I_1, \ell_H)$.

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 II-C, 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 $\mathsf{Bob}\langle\ell_H\rangle \cdot I_2\langle\ell_H\rangle\ \mathsf{sf}\ \mathsf{Bob}\langle\ell_H\rangle \cdot \mathsf{Bob}\langle\ell_H\rangle$, 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 throughout the proof. We developed the notion of compatible supercontexts in Section V-B for exactly this purpose. Indeed, if we replace Γ with an appropriate compatible supercontext when checking the potential influence of generalized principals, we remove the last barrier to a true non-interference theorem.

E. 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 $\varphi \mathbin{@} g_1$ and $\psi \mathbin{@} g_2$, if*

$$\Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g_2,$$

then either (1) $\Gamma \vdash \psi \mathbin{@} g_2$, or (2) there is some $\Delta \ll \Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g_2$, $g_1' \in \mathcal{G}(\varphi \mathbin{@} g_1)$, $g_2' \in \mathcal{G}(\psi \mathbin{@} g_2)$, 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 $\Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g_2$. For each proof rule, we argue that either $\varphi \mathbin{@} g_1$ is unnecessary for all premises or we can extend an influence from one or more subproofs to an influence from $\varphi \mathbin{@} g_1$ to $\psi \mathbin{@} g_2$. We provide details in Appendix F and are working on mechanizing the proof in Coq.

Much like other authorization logic non-interference statements [8], [26], this theorem limits when a belief $\varphi \mathbin{@} g_1$ can be necessary to prove $\psi @ g_2$ in context Γ. As we mentioned above, however, it is the first such non-interference statement for any full first-order authorization logic with discoverable trust. Moreover, it describes how FLAFOL mitigates both:

- communication between principals, via $\mathsf{canRead}$ and $\mathsf{canWrite}$ statements, and
- movement of information between security levels represented by information flow labels, via flows-to statements.

The $\mathsf{canInfl}$ 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 $\mathsf{canInfl}$ 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* $\mathsf{canInfl}$ *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 $\Gamma, \varphi @ g_1$. Similarly, if every permission that is provable in any CSC of $\Gamma, \varphi @ g_1 \vdash \psi @ g_2$ is provable under $\Gamma, \varphi @ g_1$, then we can again always instantiate Δ with $\Gamma, \varphi @ g_1$.

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—$\Gamma \vdash \varphi @ g \cdot p\langle\ell\rangle$ if and only if $\Gamma' \vdash \varphi @ g \cdot p\langle\ell\rangle$—since Theorem 5 allows us to delete all of the beliefs on which they disagree. If we think of contexts as inputs, as in a propositions-as-types interpretation, then this says that changing high inputs cannot change low results.

VII. 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 [36] and FLAM [2] 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 for reasoning about authorization-logic proofs in order to prove non-interference for FLAFOL. These reasoning principles should be expanded and used in other logics. For instance, using the tools developed in Section VI, we should be able to give non-interference proofs for logics like NAL [6] and FOCAL [37] which reason about implication and disjunction. We should also be able to add disjunction and implication to logics like DCC [8], [12] while still providing a non-interference theorem.

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 [33], [34]. 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.

VIII. Related Work

Prior work in both information flow control and authorization logics have explored connections between authorization and information security. The Decentralized Label Model [38] incorporates a notion of ownership into its information flow policies, specifying who may authorize exceptions to a policy.

The Flow-Limited Authorization Model (FLAM) [2] 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 [40] 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 [41] expresses labels in terms of principals. FLAM [2] 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 [42] explores preventing probing attacks, authorization queries which leak secret information, in Datalog-based authorization logics like DKAL [43] and SecPAL [5]. In SecPAL$^+$ [1], 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 [26] 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 [8], [12] (DCC) has been used to model information flow control and authorization, but not both. 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 [3] uses ideas from FLAM and DCC to support discoverable trust. FLAC and Polymorphic DCC [8] 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 [10], [11] 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 [11]. This setup unfortunately makes it impossible for principals to disagree about the meaning of labels, since the labels themselves define their properties.

IX. 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 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.

FLAFOL's statement of non-interference contains several subtleties. Two of these subtleties reflect the power of first-order logic, and reduce to prior non-interference assurances where those apply. However, the generalized principal construction of FLAFOL adds further complications. In particular, the non-interference statement itself is given in terms of *generalized* principals, rather than the principals themselves. We have also shown how to discuss the security of individual principals, though doing so is complicated. Interesting future work would further understand how to move results from generalized principals to the underlying principals.

References

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

[2] O. Arden, J. Liu, and A. C. Myers, "Flow-limited authorization," in *28th IEEE Symp. on Computer Security Foundations (CSF)*, Jul. 2015, pp. 569–583.

[3] O. Arden and A. C. Myers, "A calculus for flow-limited authorization," in *29th IEEE Symp. on Computer Security Foundations (CSF)*, Jun. 2016, pp. 135–147.

[4] J. Howell and D. Kotz, "A formal semantics for SPKI," in *ESORICS 2000*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1895, pp. 140–158.

[5] M. Y. Becker, C. Fournet, and A. D. Gordon, "SecPAL: Design and semantics of a decentralized authorization language," *Journal of Computer Security*, vol. 18, no. 4, pp. 619–665, 2010.

[6] F. B. Schneider, K. Walsh, and E. G. Sirer, "Nexus Authorization Logic (NAL): Design rationale and applications," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 8:1–8:28, Jun. 2011.

[7] 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)*, Oct. 1991, pp. 165–182.

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

[9] 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.

[10] 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*, Sep. 2008.

[11] L. Jia and S. Zdancewic, "Encoding information flow in AURA," *PLAS*, pp. 17–29, June 2009.
[12] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke, "A core calculus of dependency," in *26th ACM Symp. on Principles of Programming Languages (POPL)*, Jan. 1999, pp. 147–160.
[13] 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)*, Jun. 2018.
[14] L. Zheng and A. C. Myers, "End-to-end availability policies and noninterference," in *18th IEEE Computer Security Foundations Workshop (CSFW)*, Jun. 2005, pp. 272–286.
[15] J.-M. Andreoli, "Logic programming with focusing proofs in linear logic," *Journal of logic and computation*, vol. 2, no. 3, pp. 297–347, 1992.
[16] E. Z. Yang, "Logitext," 2012, accessed February 19, 2019. [Online]. Available: http://logitext.mit.edu/main
[17] J. L. Caldwell, "Classical propositional decidability via nuprl proof extraction," *TPHOLs*, 1998.
[18] D. Volpano, G. Smith, and C. Irvine, "A sound type system for secure flow analysis," *Journal of Computer Security*, vol. 4, no. 3, pp. 167–187, 1996.
[19] M. Algehed, "Short paper: A perspective on the dependency core calculus," *PLAS*, October 2018.
[20] The Coq development team, *The Coq proof assistant reference manual*, LogiCal Project, 2004, version 8.0. [Online]. Available: http://coq.inria.fr
[21] G. Takeuti, *Proof Theory*, ser. Dover Books on Mathematics. Dover Books, 1987, Second Edition, republished by Dover Books in 2013. Originally published by North-Holland, Amsterdam.
[22] J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and Types*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.
[23] F. Pfenning, "Structural cut elimination," *LICS*, pp. 156–166, June 1995.
[24] D. E. Denning, "A lattice model of secure information flow," *Comm. of the ACM*, vol. 19, no. 5, pp. 236–243, 1976.
[25] J. A. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symp. on Security and Privacy*, Apr. 1982, pp. 11–20.
[26] D. Garg and F. Pfenning, "Non-interference in constructive authorization logic," in *19th IEEE Computer Security Foundations Workshop (CSFW)*. New Jersey, USA: IEEE, 2006.
[27] S. Zdancewic and A. C. Myers, "Robust declassification," in *14th IEEE Computer Security Foundations Workshop (CSFW)*, Jun. 2001, pp. 15–23.
[28] A. Sabelfeld and A. C. Myers, "A model for delimited release," in *2003 International Symposium on Software Security*, ser. Lecture Notes in Computer Science, no. 3233. Springer-Verlag, 2004, pp. 174–191.
[29] H. Mantel and D. Sands, "Controlled Declassification based on Intransitive Noninterference," in *2nd ASIAN Symposium on Programming Languages and Systems, APLAS 2004*, ser. LNCS 3303. Taipei, Taiwan: Springer-Verlag, Nov. 2004, pp. 129–145.
[30] 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.
[31] A. Sabelfeld and D. Sands, "Dimensions and principles of declassification," in *18th IEEE Computer Security Foundations Workshop (CSFW)*, Jun. 2005, pp. 255–269.
[32] A. C. Myers, A. Sabelfeld, and S. Zdancewic, "Enforcing robust declassification and qualified robustness," *Journal of Computer Security*, vol. 14, no. 2, pp. 157–196, 2006.
[33] S. Chong and A. C. Myers, "End-to-end enforcement of erasure and declassification," in *IEEE Symp. on Computer Security Foundations (CSF)*, Jun. 2008, pp. 98–111.
[34] 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*, Sep. 2015.
[35] E. Cecchetti, A. C. Myers, and O. Arden, "Nonmalleable information flow control," in *24th ACM Conf. on Computer and Communications Security (CCS)*, Oct. 2017, pp. 1875–1891.
[36] J. Liu, O. Arden, M. D. George, and A. C. Myers, "Fabric: Building open distributed systems securely by construction," *J. Computer Security*, vol. 25, no. 4–5, pp. 319–321, May 2017.
[37] A. K. Hirsch and M. R. Clarkson, "Belief semantics of authorization logic," *CCS*, pp. 561–572, November 2013.
[38] A. C. Myers and B. Liskov, "Complete, safe information flow with decentralized labels," in *IEEE Symp. on Security and Privacy*, May 1998, pp. 186–197.
[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)*, Jul. 2006, pp. 202–216.
[40] 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*, 2008, pp. 33–47.
[41] A. C. Myers and B. Liskov, "Protecting privacy using the decentralized label model," *ACM Transactions on Software Engineering and Methodology*, vol. 9, no. 4, pp. 410–442, Oct. 2000.
[42] M. Y. Becker, "Information flow in trust management systems," *Journal of Computer Security*, vol. 20, no. 6, pp. 677–708, 2012.
[43] Y. Gurevich and I. Neeman, "DKAL: Distributed-knowledge authorization language," in *IEEE Symp. on Computer Security Foundations (CSF)*. IEEE, 2008, pp. 149–162.

APPENDIX A
THE FULL FLAFOL PROOF SYSTEM

The full FLAFOL proof system can be found in Figure 8.

APPENDIX B
SIGNED SUBFORMULAE

As we mention in Section V-A, 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 9 contains the complete rules for the *signed subformula relation* we mention in Section V-A.

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.

APPENDIX C
COMPATIBLE SUPERCONTEXTS

Figure 10 contains the full rules for compatible supercontexts.

APPENDIX D
DETAILS OF CUT ADMISSIBILITY PROOF

We provide here some details about the Coq formalization of the admissibility of the Cut rule in FLAFOL. We define FLAFOL in the file `Sequent.v`. There are three points we wish to make about this file.

First, as is suggested by Pfenning [23], we drop the structural rules from the logic (WEAKENING, EXCHANGE and CONTRACTION) and modify our rules so that they never erase

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

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

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

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

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

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

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

$$\textsc{ExistsL}\ \frac{\Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g' \qquad x \notin \mathsf{fv}(\Gamma, \psi, g, g')}{\Gamma, (\exists x\!:\!\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'}$$

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

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

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

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

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

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

Fig. 8. Full FLAFOL Proof System

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

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

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

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

$$\frac{}{(\varphi[x \mapsto t])^- \leq (\forall x \!:\! \sigma.\, \varphi)^-} \qquad \frac{}{\varphi^+ \leq (\forall x \!:\! \sigma.\, \varphi)^+}$$

$$\frac{}{\varphi^- \leq (\exists x \!:\! \sigma.\, \varphi)^-} \qquad \frac{}{(\varphi[x \mapsto t])^+ \leq (\exists x \!:\! \sigma.\, \varphi)^+}$$

$$\frac{}{\varphi^s \leq (p \ \mathsf{says}_\ell\ \varphi)^s}$$

Fig. 9. Signed Subformula Relation

anything from the context. This makes meta-theoretic proofs simpler and we prove that the removed rules are admissible.

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 deal with features that we left for future work, but we don't 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, due to some of Coq's shortcomings 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.

In the file `NormalForm.v` we define a normal form for FLAFOL proofs. The cut-elimination procedure uses normalization into this form as an essential step. We say that a proof is in First Normal Form when as soon as a rule which manipulates something other than a formula is used, the rest of the proof is in Second Normal Form. Similarly, we say that a proof is in Second Normal Form if it never uses logical rules (except TRUER and FALSEL). For instance, if a proof in First Normal Form uses the VARR rule, it can't use ANDR anymore. This notion of FLAFOL Normal Form proof should not be confused with "normal proof" in the literature which usually means cut-free. The main theorem proven in this file is that every provable sequent has a First Normal Form.

Lastly the file `Cut.v` contains the cut-elimination procedure. Its structure is inspired by Pfenning's structural cut elimination [23]: nested triple induction on the formula being cut and on both proofs. Due to the presence of rules like VARR and FWDR, this exact proof strategy sometimes fails. Instead, we first normalize the proofs for $\Gamma \vdash \varphi \mathbin{@} g$ and $\Gamma, \varphi \mathbin{@} g \vdash \psi \mathbin{@} g'$. If they're both in First Normal Form but not in Second Normal Form, we proceed as Pfenning suggests. If, however, one of them is in Second Normal Form we define 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 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`

APPENDIX E
EXAMPLES OF PERMISSION MODELS

We now show how FLAFOL can encode multiple concrete permission models. This demonstrates the expressive power of FLAFOL's permission system and helps show ways in which it might be used.

There is no particular reason for there to be some external model of permissions. The "default" permission model simply gives meaning to $\mathsf{canRead}$ and $\mathsf{canWrite}$ through their behavior.

That is, the only properties of $\mathsf{canRead}$ and $\mathsf{canWrite}$ are CRVAR, CWVAR, and the assumptions we make about them in the form of FLAFOL formulae. This is appropriate in many cases. For instance, in the example of viewing photos on social media, $\mathsf{canRead}$ and $\mathsf{canWrite}$ 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, $\mathsf{canWrite}$ can have a trivial implementation:

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

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 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 confidentiality and integrity to be the only part of principals/labels that matter.

We can formalize this by giving projection functions $\pi_{P,C}, \pi_{P,I}, \pi_{L,C}, \pi_{L,I}$. The subscripts on the names of these functions tell us what data they operate on. If the first character is a P, it takes a principal as its argument; if the first character is an L it takes a label. If the second character is a C, it provides confidentiality label as its output; if the first character is an I it provides an integrity label. 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

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

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

$$\textsc{CSCAndR1}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \wedge \psi \mathbin{@} g} \qquad \textsc{CSCAndR2}\ \frac{\Delta \ll \Gamma \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \wedge \psi \mathbin{@} g} \qquad \textsc{CSCAndL}\ \frac{\Delta \ll \Gamma, (\varphi \mathbin{@} g), (\psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \wedge \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}$$

$$\textsc{CSCOrR1}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \vee \psi \mathbin{@} g} \qquad \textsc{CSCOrR2}\ \frac{\Delta \ll \Gamma \vdash \psi \mathbin{@} g}{\Delta \ll \Gamma \vdash \varphi \vee \psi \mathbin{@} g} \qquad \textsc{CSCOrL1}\ \frac{\Delta \ll \Gamma, \varphi \mathbin{@} g \vdash \chi \mathbin{@} g'}{\Delta \ll \Gamma, (\varphi \vee \psi \mathbin{@} g) \vdash \chi \mathbin{@} g'}$$

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

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

$$\textsc{CSCForallR}\ \frac{\Delta \ll \Gamma \vdash \varphi \mathbin{@} g \qquad x \notin \mathsf{fv}(\Gamma, g)}{\Delta \ll \Gamma \vdash \forall x{:}\sigma.\, \varphi \mathbin{@} g} \qquad \textsc{CSCForallL}\ \frac{\Delta \ll \Gamma, \varphi[x \mapsto t] \mathbin{@} g \vdash \psi \mathbin{@} g'}{\Delta \ll \Gamma, (\forall x{:}\sigma.\, \varphi \mathbin{@} g) \vdash \psi \mathbin{@} g'}$$

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

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

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

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

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

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

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

Fig. 10. Compatible Supercontext Rules

$\pi_{L,I}(\ell)$ is "the integrity component of label ℓ." With these functions, we can say that

$$p \;\mathsf{says}_\ell\; \mathsf{canRead}(q,\ell) \leftrightarrow p \;\mathsf{says}_\ell\; (\pi_{L,C}(\ell) \sqsubseteq \pi_{P,C}(q)),$$
and
$$p \;\mathsf{says}_\ell\; \mathsf{canWrite}(q,\ell) \leftrightarrow p \;\mathsf{says}_\ell\; (\pi_{P,I}(q) \sqsubseteq \pi_{L,I}(\ell)).$$

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

FLAFOL can also encode *capabilities*. A capability is a token which can be presented to the owner of some data to gain read or write access to that data. We consider copyable, delegatable tokens with a single root authority. That is, when a principal owns a token, they can copy that token and give the copy to another principal. The root authority is a principal who may forge new tokens freely. We represent a principal p having a read token for the label ℓ using the relation $\mathsf{hasRToken}(p,\ell)$.

We state that q is a root authority if, for every p,

$$\forall r\!:\!\mathsf{principal}.\begin{pmatrix} q \;\mathsf{says}_\ell\; \mathsf{hasRToken}(r,\ell) \to \\ \quad p \;\mathsf{says}_\ell\; \mathsf{hasRToken}(r,\ell) \end{pmatrix}$$

This says that q gets to forge new tokens and give them away, and every other principal will accept this token. To say that tokens are copyable and delegatable is to say that for every p,

$$\forall q\!:\!\mathsf{principal}.\, p \;\mathsf{says}_\ell\; \mathsf{hasRToken}(q,\ell) \to \begin{pmatrix} \forall r\!:\!\mathsf{principal}. \\ \begin{pmatrix} q \;\mathsf{says}_\ell\; \mathsf{hasRToken}(r,\ell) \to \\ \quad p \;\mathsf{says}_\ell\; \mathsf{hasRToken}(r,\ell) \end{pmatrix} \end{pmatrix}$$

This says that if q holds a token to read ℓ, then they can forge a new token and give it to r, and every other principal will accept this token.

To connect these to FLAFOL's permission relations, we simply say that holding a token provides read permissions. That is, for every principal p,

$$p \;\mathsf{says}_\ell\; \mathsf{hasRToken}(q,\ell) \leftrightarrow p \;\mathsf{says}_\ell\; \mathsf{canRead}(q,\ell).$$

The formulae above are enough to model copyable, delegatable read capabilities with a single root authority. Note that while we have only shown read capabilities, the write capabilities formulae are almost identical.

Appendix F
Proof of Non-Interference

We now provide a proof of our non-interference statement. First, we prove two CSC rules are admissible.

Lemma 1. *The following two rules are admissible up to α equivalence of $\Gamma \vdash \varphi \;@\; g_1$:*

$$\textsc{CSCAtomic}\;\frac{\Delta \ll \Gamma \vdash \varphi \;@\; g_1 \qquad \varphi \text{ is atomic}}{\Delta \ll \Gamma \vdash \psi \;@\; g_2}$$

$$\textsc{CSCWeaken}\;\frac{\Delta \ll \Gamma \vdash \varphi \;@\; g_1}{\Delta, \psi \;@\; g_2 \ll \Gamma, \psi \;@\; g_2 \vdash \varphi \;@\; g_1}$$

Proof. Both proofs are by induction on the derivation of $\Delta \ll \Gamma \vdash \varphi \;@\; g_1$.

CSCAtomic follows from the fact that no right rules that consider the shape of φ apply when it is atomic. CSCIR, CSCSCR, CSCVarR, and CSCFwdR can simply be eliminated when we replace $\varphi \;@\; g_1$ with $\psi \;@\; g_2$. All other rules either cannot apply when φ is atomic, or apply equally with $\psi \;@\; g_2$.

CSCWeaken follows from a direct straightforward induction (with heavy use of CSCExchange) noting that CSCRefl is the only way to terminate the induction. □

Theorem 5 (Non-Interference)**.** For all contexts Γ and beliefs $\varphi \;@\; g_1$ and $\psi \;@\; g_2$, 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 $\Delta \vdash g_1' \cdot g_1'' \;\mathsf{canInfl}\; g_2'$.

Proof. This is a proof by induction on the proof of $\Gamma, \varphi \;@\; g_1 \vdash \psi \;@\; g_2$. As part of our induction, we claim that if (1) holds than the proof of $\Gamma \vdash \psi \;@\; g_2$ is no larger than original proof. This means that when an inductive application produces the first case, we can safely apply the inductive hypothesis again to the result.

Most of the cases are straightforward. We illustrate a typical right rule with AndR and a typical left rule with OrL. We then handle the other cases of note.

Case AndR: Here we consider the rule

$$\frac{\Gamma, \varphi \;@\; g_1 \vdash \psi_1 \;@\; g_2 \qquad \Gamma, \varphi \;@\; g_1 \vdash \psi_2 \;@\; g_2}{\Gamma, \varphi \;@\; g_1 \vdash \psi_1 \wedge \psi_2 \;@\; g_2}$$

We start by applying our inductive hypothesis to the left premise. If condition (2) holds, we know that any compatible supercontext of that premise is also a compatible supercontext of the proven sequent by CSCAndR1, and $\mathcal{G}(\psi_1 \;@\; g_2) \subseteq \mathcal{G}(\psi_1 \wedge \psi_2 \;@\; g_2)$ by the definition of $\mathcal{G}$. Therefore the same generalized principals prove (2). If (1) holds we apply our inductive hypothesis to the right premise. Again, if (2) holds there, we similarly acquire the necessary influence for (2). Otherwise we can remove $\varphi \;@\; g_1$ from both premises and prove (1) with AndR.

Case OrL: Here we consider the rule

$$\frac{\Gamma, \chi_1 \;@\; g_3 \vdash \psi \;@\; g_2 \qquad \Gamma, \chi_2 \;@\; g_3 \vdash \psi \;@\; g_2}{\Gamma, (\chi_1 \vee \chi_2 \;@\; g_3) \vdash \psi \;@\; g_2}$$

As this is a left rule, we must consider two cases: one where $\varphi \;@\; g_1$ is the active belief (in this case $\chi_1 \vee \chi_2 \;@\; g_3$) and one where it is not.

When $\varphi \;@\; g_1$ is the active belief, we simply apply our inductive hypothesis to the left premise. If (1) inductively holds, then that is exactly a proof of (1) for the main sequent. If (2) inductively holds, then CSCOrL1 means we can use the same Δ, and since $\mathcal{G}(\chi_1 \;@\; g_3) \subseteq \mathcal{G}(\chi_1 \vee \chi_2 \;@\; g_3)$, the same generalized principals result in the necessary influence to prove (2).

When $\varphi \;@\; g_1$ is not the active belief, we apply the inductive hypothesis to both premises. If we can remove the belief from the context in both, the we can use OrL to prove (1). Otherwise we directly construct the necessary influence to prove (2).

Case ForallR: Here we consider the rule

$$\frac{\Gamma, \varphi \mathbin{@} g_1 \vdash \chi \mathbin{@} g_2 \qquad x \notin \mathsf{fv}(\Gamma, g)}{\Gamma, \varphi \mathbin{@} g_1 \vdash \forall x{:}\sigma.\, \chi \mathbin{@} g_2}$$

We apply our inductive hypothesis. If (1) holds inductively, we can prove (1) by re-applying ForallR. If (2) holds, we note that x is a valid term of sort σ, so $\mathcal{G}(\chi \mathbin{@} g_2) \subseteq \mathcal{G}(\forall x{:}\sigma.\, \chi \mathbin{@} g_2)$, meaning the influence from the inductive application proves (2) as desired.

Case SaysR: Here we consider the rule

$$\frac{\Gamma, \varphi \mathbin{@} g_1 \vdash \chi \mathbin{@} g_2 \cdot p\langle\ell\rangle}{\Gamma, \varphi \mathbin{@} g_1 \vdash p \;\mathsf{says}_\ell\; \chi \mathbin{@} g_2}$$

We again apply our inductive hypothesis to the premise. If we can remove the context belief in the premise, then SaysR proves (1). If we find an influence, $\mathcal{G}(\chi \mathbin{@} g_1 \cdot p\langle\ell\rangle) = \mathcal{G}(p \;\mathsf{says}_\ell\; \chi \mathbin{@} g_2)$ by definition, and CSCSaysR says that the necessary compatible supercontext is valid, again meaning the same influence given by the inductive hypothesis proves (2).

Case ImpL: Here we consider use of the rule

$$\frac{\Gamma \vdash \chi_1 \mathbin{@} \langle\rangle \qquad \Gamma, \chi_2 \mathbin{@} g_3 \vdash \psi \mathbin{@} g_2}{\Gamma, (\chi_1 \to \chi_2 \mathbin{@} g_3) \vdash \psi \mathbin{@} g_2}$$

Again, as a left rule, we separately consider when $\varphi \mathbin{@} g_1$ is the active belief and when it is not.

When it is the active belief, we simply apply the inductive hypothesis to the right premise. If (1) holds inductively, then $\Gamma \vdash \psi \mathbin{@} g_2$ is provable and we are done. If (2) holds inductively, then CSCImpL1 allows us to use the same Δ, and since $\mathcal{G}(\chi_2 \mathbin{@} g_3) = \mathcal{G}(\chi_1 \to \chi_2 \mathbin{@} g_3)$ by definition, the same generalized principals create the necessary influence.

When $\varphi \mathbin{@} g_1$ is not the active belief, this case is more complicated. First we apply our inductive hypothesis to the right premise. If (2) holds, we are done as the influence remains the same. If (1) holds, we have $\Gamma', (\chi_2 \mathbin{@} g_3) \vdash \psi \mathbin{@} g_2$ where $\Gamma', \varphi \mathbin{@} g_1 = \Gamma$.

We now apply our inductive hypothesis again to this resulting proof, thus checking if we can remove $\chi_2 \mathbin{@} g_3$. If we can remove that assumption, then Weakening proves (1). Otherwise, there is some $g_3' \in \mathcal{G}(\chi_2 \mathbin{@} g_3)$, $g_2' \in \mathcal{G}(\psi \mathbin{@} g_2)$, g_3'', and some $\Delta_1 \ll \Gamma', \chi_2 \mathbin{@} g_3 \vdash \psi \mathbin{@} g_2$ such that

$$\Delta_1 \vdash g_3' \cdot g_3'' \;\mathsf{canInfl}\; g_2'.$$

We make a third use of our inductive hypothesis, this time applying to the original left premise $\Gamma \vdash \chi_1 \mathbin{@} \langle\rangle$. If (1) holds inductively, then we have $\Gamma' \vdash \chi_1 \mathbin{@} \langle\rangle$, and reapplying ImpL proves (1). If (2) inductively holds, that means there is some $g_1' \in \mathcal{G}(\varphi \mathbin{@} g_1)$, $g \in \mathcal{G}(\chi_1 \mathbin{@} \langle\rangle)$, g_1'', and some $\Delta_2 \ll \Gamma \vdash \chi_1 \mathbin{@} \langle\rangle$ such that

$$\Delta_2 \vdash g_1' \cdot g_1'' \;\mathsf{canInfl}\; g.$$

By CSCImpL2, Δ_2 is a CSC of the original sequent, and by CSCImpL1 and CSCWeaken so is $\Delta_1, \varphi \mathbin{@} g_1$. Thus by CSCUnion, if we let $\Delta = (\Delta_1, \varphi \mathbin{@} g_1) \cup \Delta_2$, then $\Delta \ll \Gamma, (\chi_1 \to \chi_2 \mathbin{@} g_3) \vdash \psi \mathbin{@} g_2$. Finally, since $\chi_1 \to \chi_2 \mathbin{@} g_3 \in \Delta$, ImpCI says

$$\Delta \vdash g \;\mathsf{canInfl}\; g_3'.$$

Using ExtCI and TransCI, we therefore have

$$\Delta \vdash g_1' \cdot g_1'' \cdot g_3'' \;\mathsf{canInfl}\; g_2'.$$

Letting $g_1'' \cdot g_3''$ be the relevant extension, this proves (2).

Case FwdR: Here we consider the rule

$$\frac{\begin{array}{c}\Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g' \\ \Gamma, \varphi \mathbin{@} g_1 \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Gamma, \varphi \mathbin{@} g_1 \vdash \mathsf{canWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Gamma, \varphi \mathbin{@} g_1 \vdash \psi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g'}$$

We begin by applying our inductive hypothesis to the premise $\Gamma, \varphi \mathbin{@} g_1 \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle$. If (2) holds, then there is some $g_1' \in \mathcal{G}(\varphi \mathbin{@} g_1)$, $g' \in \mathcal{G}(\mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle)$, g_1'', and some $\Delta \ll \Gamma, \varphi \mathbin{@} g_1 \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle$ such that

$$\Delta \vdash g_1' \cdot g_1'' \;\mathsf{canInfl}\; g'.$$

However, $\mathcal{G}(\mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle) = \{g \cdot p\langle\ell\rangle\}$, and therefore $\Delta \vdash g_1' \cdot g_1'' \;\mathsf{canInfl}\; g \cdot p\langle\ell\rangle$. Since $\Gamma, (\varphi \mathbin{@} g_1) \subseteq \Delta$, we have

$$\textsc{FwdSF}\ \frac{\begin{array}{c}\Delta \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Delta \vdash \mathsf{canWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Delta \vdash g \cdot p\langle\ell\rangle \;\mathsf{sf}\; g \cdot q\langle\ell\rangle}$$

Therefore by SF-CI and TransCI, $\Delta \vdash g_1' \cdot g_1'' \;\mathsf{canInfl}\; g \cdot q\langle\ell\rangle$. Since all elements of $\mathcal{G}(\psi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g')$ are extensions of $g \cdot q\langle\ell\rangle$ and Δ is a compatible supercontext of the final sequent by CSCAtomic, this means we can extend by some g_1''' and use ExtCI to prove that, for every $g_2' \in \mathcal{G}(\psi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g')$,

$$\Delta \vdash g_1' \cdot (g_1'' \cdot g_1''') \;\mathsf{canInfl}\; g_2'.$$

This proves (2).

If no such Δ, g_1', g_1'' exist (for the original influence), then clearly there can be no influence to either $g \cdot q\langle\ell\rangle$ or any extension of $g \cdot p\langle\ell\rangle \cdot g'$. This means, by induction, we must be able to prove all three premises without $\varphi \mathbin{@} g_1$ in the context, allowing us to prove (1) by re-applying FwdR.

Case FwdL: Here we consider the rule

$$\frac{\begin{array}{c}\Gamma, (\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g_2 \\ \Gamma, (\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle \\ \Gamma, (\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{canWrite}(p, \ell) \mathbin{@} g \cdot q\langle\ell\rangle\end{array}}{\Gamma, (\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g_2}$$

As this is a left rule, we must separately consider whether or not $\varphi \mathbin{@} g_1$ is the active belief.

As above, we begin with the case where $\varphi \mathbin{@} g_1$ is active. In this case we start by applying our inductive hypothesis to the premise $\Gamma, (\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g_2$. If (1) is inductively true, that proves (1) in this case. Otherwise there is some

$g_3 \in \mathcal{G}(\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g')$, some $g_2' \in \mathcal{G}(\psi \mathbin{@} g_2)$, some g_3', and some compatible supercontext Δ such that

$$\Delta \vdash g_3 \cdot g_3' \ \mathsf{canInfl}\ g_2'.$$

Note that we know that $g_3 = g \cdot q\langle\ell\rangle \cdot g''$ for some g''. By FwdSF, we know that $\Delta \vdash g \cdot p\langle\ell\rangle \ \mathsf{sf}\ g \cdot q\langle\ell\rangle$. Therefore, by repeated application of ExtSF, we can prove

$$\Delta \vdash g \cdot p\langle\ell\rangle \cdot g'' \cdot g_3' \ \mathsf{sf}\ g \cdot q\langle\ell\rangle \cdot g'' \cdot g_3'.$$

We additionally note that $g \cdot q\langle\ell\rangle \cdot g'' \in \mathcal{G}(\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g')$, so therefore $g_1' = g \cdot p\langle\ell\rangle \cdot g'' \in \mathcal{G}(\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g')$. Thus by TransCI, we have that

$$\Delta \vdash g_1' \cdot g_3' \ \mathsf{canInfl}\ g_2'.$$

By CSCFwdL, Δ is a compatible supercontext of the proven sequent, so this proves (2).

We now consider the case where $\varphi \mathbin{@} g_1$ is not the active belief. In this case we begin with another application of the inductive hypothesis on the first premise. If (2) holds, then the influence directly proves (2). Otherwise we have $\Gamma', (\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g_2$ where $\Gamma', \varphi \mathbin{@} g_1 = \Gamma$. We now apply our inductive hypothesis to the second premise $(\Gamma, (\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g') \vdash \mathsf{canRead}(q, \ell) \mathbin{@} g \cdot p\langle\ell\rangle)$. If we can remove $\varphi \mathbin{@} g_1$ from the context, then by the same logic as the previous case, we can do the same for the third premise and prove (1) by re-applying FwdL. Otherwise there is some $g_1' \in \mathcal{G}(\varphi \mathbin{@} g_1)$, g_1'', and some $\Delta_1 \ll \Gamma, (\chi \mathbin{@} g \cdot p\langle\ell\rangle \cdot g')$ such that

$$\Delta_1 \vdash g_1' \cdot g_1'' \ \mathsf{canInfl}\ g \cdot p\langle\ell\rangle.$$

For our last use of the inductive hypothesis, we apply it to the stripped-down first premise, in that case trying to remove the belief $\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g'$. If we can remove belief, we can prove (1) using Weakening. Otherwise, there is some $\Delta_2 \ll \Gamma', (\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g') \vdash \psi \mathbin{@} g_2$, and some $g_3 \in \mathcal{G}(\chi \mathbin{@} g \cdot q\langle\ell\rangle \cdot g')$, $g_2' \in \mathcal{G}(\psi \mathbin{@} g_2)$, and g_3' such that

$$\Delta_2 \vdash g_3 \cdot g_3' \ \mathsf{canInfl}\ g_2'.$$

Letting $\Delta = \Delta_1 \cup (\Delta_2, \varphi \mathbin{@} g_1)$, we know it is a valid compatible supercontext and proves the same influences as in the ImpL case above. Additionally, as in the FwdR case above, we know that $\Delta \vdash g \cdot p\langle\ell\rangle \ \mathsf{sf}\ g \cdot q\langle\ell\rangle$ and therefore there is some g_1''' such that

$$\Delta \vdash g_1' \cdot (g_1'' \cdot g_1''') \ \mathsf{canInfl}\ g_3 \cdot g_3'.$$

Therefore TransCI proves the influence needed for (2). This completes the most complex case of the proof.

The VarR and VarL cases are slight simplifications of FwdR and FwdL, respectively. All other rules are straightforward. □