Proof Search in an Authorization Logic

Deepak Garg

Published on April 14, 2009
Revised on September 20, 2009
CMU-CS-09-121

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

e-mail: dg@cs.cmu.edu

Abstract

We consider the problem of proof search in an expressive authorization logic that contains a "says" modality and an ordering on principals. After a description of the proof system for the logic, we identify two fragments that admit complete goal-directed and saturating proof search strategies. A smaller fragment is then presented, which supports both goal-directed and saturating search, and has a sound and complete translation to first-order logic. We conclude with a brief description of our implementation of goal-directed search.

This work was supported partially by the iCAST project sponsored by the National Science Council, Taiwan, under grant no. NSC97-2745-P-001-001, and partially by the Air Force Research Laboratory under grant no. FA87500720028.

Keywords: Access control, Logic Programming, Proof search

1 Introduction

In the recent past, the use of logic to represent, enforce, and reason about access control policies has grown significantly. Several systems and programming languages have been proposed that express policies as logical theories, and allow or deny access depending on the provability of specific formulas (e.g., [8–10, 21, 27, 35, 37, 53]). In all these applications efficient methods of proof search are required. When human intervention is not possible or undesirable the methods need to be automated. In this paper we develop the theory of logic programming in a logic designed for expressing access control policies (such logics are generically called authorization logics here). Our primary motivation is to study the theoretical foundations of proof search in the context of access control. The issue of theoretical foundations has been ignored by prior work in the area, most of which is guided by concerns like minimizing network usage and human interaction during distributed proof search [10, 11, 22, 41]. We consider logic programming instead of full theorem proving since logic programming can be implemented efficiently. Existing work on policy languages already shows that most, if not all, policies of practical interest fit nicely into the logic programming paradigm.

Logic programming with authorization logics is non-trivial due to the inclusion of a family of modalities, commonly written k says s [3–5, 8, 9, 17, 24, 30, 31, 33]. k says s means that principal k states or believes formula s and is used to distinguish statements of different principals. While the modality makes it easier to express decentralized policies, it significantly increases the complexity of proof search. Further, there is a trade-off between choosing a proof-theoretic interpretation of k says s that is useful for expressing for practical policies, and one that is amenable to proof search, especially through logic programming.

Based on these considerations, we introduce a new authorization logic, BL_0, that is not only expressive but also well suited to logic programming. After describing its proof system and commenting briefly on its expressiveness, we examine two different strategies for finding proofs in it: (a) Goal-directed search (backward chaining), which is often efficient for finding large proofs of authorization one at a time, and (b) Saturating search (forward chaining), which is useful for determining all possible permissions that follow from a policy. For each of the two search strategies, we identify a fragment of BL_0 on which the strategy is complete with respect to the logic's proof system. We then identify a third fragment to which both strategies apply (via a translation). We also present a complete translation from this third fragment to first-order intuitionistic logic, thus opening the possibility of using existing theorem provers for BL_0. A minor contribution of this paper is a justification for the semantics of the policy language Binder [23], and a formalization of its connection to a related language Soutei [49].

Before proceeding to the technical material, we would like to clarify the scope of this work and its broad context. This work should be distinguished from policy languages and trust management systems that have implementations based on ideas from or translations to logic programs [12, 13, 23, 36, 38–40, 49]. Despite their importance in practice, the inference rules of these frameworks lack an internal justification that is readily available in a logic in the form of meta-theoretic principles like cut-elimination. One goal of this

paper is to combine such meta-theoretic foundations with efficient implementations in an authorization logic.

BL_0 is a simplification of a larger logic BL, which includes support for explicit time [24], constraints, and system state, all of which BL_0 lacks. Owing to the additional complexity, logic programming in BL is even harder than it is in BL_0, and an attempt to extend the results of this paper to the full logic is currently underway. The board goal of this project is to use BL for expressing and enforcing access policies in our implementation of a file system based on proof-carrying authorization [8–10]. For this application, we have already implemented the goal-directed search strategy described in this paper in a user tool that is available online [28]. In addition to the file system, we also intend to use BL in a type system and a compiler that translates policies expressed in BL to file system permissions. For the latter, we expect that saturating search may be the best choice.

Organization. The rest of this paper is organized as follows. After discussing closely related work, we introduce the logic BL_0, its sequent calculus, and some meta-theorems including cut-elimination (Section 2). In Sections 3 and 4 we discuss goal-directed search and saturating search respectively, and identify fragments of BL_0 over which the two search strategies are complete. Section 5 introduces the third fragment that allows both forms of search, and presents its translation to first-order logic. Section 6 discusses some open issues and briefly comments on our implementation of goal-directed search. Section 7 concludes the paper.

Related Work. Perhaps most closely related to our work in spirit are the policy languages Binder and Soutei [23, 49], whose says modality and search strategies were the inspiration for this work. (The name BL stands for Binder Logic as a tribute to the inspiration.) However, we should carefully observe the distinction here: Binder and Soutei are domain specific languages whose rules have no particular logical justification whereas BL_0 is a logic. We do show in Section 5 that both Binder and Soutei are closely related to a fragment of BL_0. Along the same lines, there are several other policy languages and trust management systems that implement or admit search procedures, often by translation to logic programs (e.g., [12–16, 32, 36, 38–40]).

Also related is work on construction of proofs for access when credentials are distributed on a network [10, 11, 22, 41]. Our setting is quite different. We assume local access to credentials so that issues like credential chain discovery and minimizing network usage are irrelevant to our cause. On the other hand, we expect to work with large policies and use proof construction for system interfaces, so we need to address the problem of constructing proofs very fast (at the order of milliseconds).

BL_0 is one of many authorization logics [3–5, 8, 9, 17, 24, 30, 31, 33] (see [2] for a survey) but its says modality is different from that in any of the these logics. BL_0's says modality is based on the necessitation modality of the modal logic CS4 [6, 46]. The sequent calculus for BL_0 draws on ideas from a judgmental presentation of CS4 due to Pfenning and Davies [46]. An earlier paper by the author formalizes the connection between BL_0 and

CS4 ([29]; Section 5.5).

There is a large body of work on logic programming with modalities (see [44] for a survey of the area). However, our work is not directly related to any of these. Instead, our treatment of goal-directed search extends prior work on uniform proofs for first-order logic [43] and draws on ideas from the linear logic programming language Lolli [34]. BL_0's saturating search is motivated by Datalog (see e.g., [18]), the Concurrent Logical Framework [52] and the logic programming language LolliMon [42]. Both search principles are related to and build upon the general principle of focusing [7]. Our proof techniques are based on a lot of earlier work that is cited throughout the paper.

2 BL_0: An Authorization Logic

BL_0 extends first-order intuitionistic logic with a modality k says s, which means that principal k states, claims, or believes that formula s is true. Predicates P express relations between terms that are either ground constants a, bound variables x, or applications of uninterpreted function symbols f to ground terms. Terms are classified into sorts σ (sometimes called types). We stipulate at least one sort principal whose elements are represented by the letter k. We also assume a preorder on ground principals $k \succeq k'$, read k is stronger than k', with a distinguished strongest element ℓ ($\ell \succeq k$ for every k). ℓ represents the "local authority" at the point of access [1]. Formulas s may either be atomic (p, q) or they may be constructed using the usual connectives of predicate logic and the special connective k says s.

Sorts	σ	::=	$\mathsf{principal} \mid \ldots$
Terms	t, k, i	::=	$a \mid x \mid f(t_1, \ldots, t_n) \mid \ell$
Predicates	P		
Atoms	p, q	::=	$P(t_1, \ldots, t_n)$
Formulas	r, s	::=	$p \mid r \wedge s \mid r \vee s \mid r \supset s \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid \exists x{:}\sigma.s \mid k \mathsf{\ says\ } s$

Axiomatic proof system. The primary proof system that we need for proof search is a sequent calculus. However before presenting that, we briefly describe an axiomatic proof system for BL_0. This proof system is obtained by extending any axiomatization of first-order intuitionistic logic with the following axioms and rules for says.

$$\frac{\vdash s}{\vdash k \mathsf{\ says\ } s} \quad \text{(N)}$$

$$\vdash (k \mathsf{\ says\ } (s_1 \supset s_2)) \supset ((k \mathsf{\ says\ } s_1) \supset (k \mathsf{\ says\ } s_2)) \quad \text{(K)}$$

$$\vdash (k \mathsf{\ says\ } s) \supset k' \mathsf{\ says\ } k \mathsf{\ says\ } s \quad \text{(I)}$$

$$\vdash k \mathsf{\ says\ } ((k \mathsf{\ says\ } s) \supset s) \quad \text{(C)}$$

$$\vdash (k' \mathsf{\ says\ } s) \supset k \mathsf{\ says\ } s \text{ if } (k' \succeq k) \quad \text{(S)}$$

Rule (N) means that each principal states at least all tautologies. Axiom (K) means that the statements of each principal are closed under implication. Together they imply that

each (k says $\cdot$) is a *normal* modality (see e.g., [26]). Axiom (I) was first suggested in the context of access control by Abadi [2]. It means that statements of any principal k can be injected into the belief system of any another principal. Axiom (C) or *conceit* states that every principal k believes that each of its statements is true. (S) means that statements of each principal are believed by all weaker principals. In particular, (ℓ says s) $\supset$ k says s for each k and s.

This choice of axioms for says is quite different from any of the existing authorization logics, and has been developed to make logic programming simpler.

Expressiveness. Besides being well-suited to logic programming, BL_0 is also quite expressive. Although there is no quantitative metric to measure the expressiveness of an authorization logic, we list below two connections between existing authorization frameworks and BL_0, that we believe are sufficient to establish the usefulness of BL_0 as a logic for writing access policies.

- Authorization logics that interpret says as a monad, referred to by many different names including CDD and ICL [3, 30, 31], can be translated in a sound and complete manner to BL_0. (See an earlier paper for details [29].) These authorization logics have been used in many applications, including type systems [27, 35, 37].

- The policy language Binder [23] is closely related to a small fragment of BL_0 and another policy language, Soutei [49], is a subset of it (see Section 5 for details). As a result any policies expressible in these languages can be written in BL_0. Soutei has been deployed in at least one large application: a publish-subscribe web service with distributed and compartmentalized administration.

We have also completed a significant case study that uses BL_0 to formalize policies for control and dissemination of classified information in USA. This case study is available online [28]. We introduce here a simplified fragment of this case study that we use as a running example in this paper.

Example 1. Consider a hypothetical intelligence agency where each file and each individual is assumed to have a classification level, which is an element of the ordered set confidential $<$ secret $<$ topsecret. Access control for files uses three distinguished principals: admin who makes final decisions on granting access, system who is responsible for governing files (e.g., setting their ownership and classification levels), and hr who is responsible for governing individuals (e.g., giving them classification levels and employment certifications). Figure 1 shows the policies that control access (numbered (1)–(5)) as well as some additional assumptions needed to get access in a specific case (numbered (6)–(10)).

In order that principal k may read file f, the following formula must be established from the policies in effect: admin says $\mathtt{may}(\mathsf{read}, k, f)$. This is possible if k is an employee of the intelligence organization (predicate $\mathtt{employee}(k)$), k has a classification level above the file (predicate $\mathtt{hasLevelForFile}(k, f)$), and k gets permission from the owner of the file. This

Common policies:

admin says $\forall k, k', f.$

(1) $(((\mathsf{hr\ says\ employee}(k)) \wedge (\mathtt{hasLevelForFile}(k, f)) \wedge$
$(\mathsf{system\ says\ owns}(k', f))(k'\ \mathsf{says\ may}(\mathsf{read}, k, f))) \supset \mathsf{may}(\mathsf{read}, k, f))$

admin says $\forall k, f, l, l'.$

(2) $(((\mathsf{system\ says\ levelFile}(f, l)) \wedge (\mathsf{hr\ says\ levelPrin}(k, l')) \wedge$
$(\mathtt{below}(l, l'))) \supset \mathtt{hasLevelForFile}(k, f))$

(3) ℓ says below(confidential, secret)

(4) ℓ says below(secret, topsecret)

(5) ℓ says below(confidential, topsecret)

Additional policies for example scenario:

(6) system says levelFile(secret.txt, secret)

(7) system says owns(Alice, secret.txt)

(8) hr says employee(Bob)

(9) hr says levelPrin(Bob, topsecret)

(10) Alice says may(read, Bob, secret.txt)

Figure 1: Simplified policies for control of classified information

is captured in policy (1) which is created by admin. For readability, we omit all sort annotations from quantifiers. Policy (2) defines the predicate $\mathtt{hasLevelForFile}(k, f)$ further in terms of classification levels of k and f (predicates $\mathtt{levelPrin}(k, l)$ and $\mathtt{levelFile}(f, l)$ respectively). The predicate $\mathtt{below}(l, l')$ captures the order $l < l'$ between classification levels (policies (3)–(5)). Since we assume that all principals believe this order, policies (3)–(5) are stated by the strongest principal ℓ.

As an illustration of the use of these policies, let us assume that file secret.txt owned by Alice is classified at the level secret. Suppose that Bob is an employee cleared at level topsecret , and further that Alice wants to let Bob read file secret.txt. We may capture this information in formulas (6)–(10). Then it is easy to see that (1)–(10) entail admin says may(read, Bob, secret.txt) using the axioms of BL_0. The non-standard axioms (I) and (S) play a significant role in this derivation.

In practice, the assumptions or policies (1)–(10) would be established using digitally signed certificates, signers being the principals who state the respective policies. For example, the certificate establishing assumption (1) would be signed by admin. The proof which shows that (1)–(10) establish admin says may(read, Bob, secret.txt) can be used as evidence that Bob has legitimate access to secret.txt. Depending on the mechanism of policy enforcement, this proof may either be checked by a reference monitor prior to access as in proof-carrying authorization [8, 9], or it may be logged for subsequent audit [51], or it may be compiled to lower level permissions.

2.1 Sequent Calculus

Next, we develop a sequent calculus for BL_0, which we later use as the basis for proof search. We follow the judgmental method [19, 46], introducing a separate category of judgments that are established by deductions. For BL_0, we need two judgments: s true meaning that formula s is provable in the current context, and k claims s meaning that principal k believes that s must be the case. (k claims s) is logically equivalent to (k says s) true. We often abbreviate s true to s. Using the judgmental method makes the technical development easier, especially the proof of cut-elimination.

$$\begin{array}{llll} \text{Judgments} & J & ::= & s \text{ true} \mid k \text{ claims } s \\ \text{Sorting} & \Sigma & ::= & a_1{:}\sigma_1 \ldots a_n{:}\sigma_n \\ \text{Hypotheses} & \Gamma & ::= & J_1 \ldots J_n \qquad (n \geq 0) \\ \text{Sequent} & & & \Sigma; \Gamma \xrightarrow{k} s \text{ true} \end{array}$$

Sequents in BL_0 have the form $\Sigma; \Gamma \xrightarrow{k} s$ true. Σ is a map from term constants occurring in the sequent to their sorts, and Γ is the set of assumed judgments (hypotheses). The novelty here is the principal k on the sequent symbol, which we call the *context* of the sequent.[1] The context represents the principal relative to whose beliefs the reasoning is being performed. It affects provability in the following manner: while reasoning in context k, an assumption of the form k' claims s entails s true if $k' \succeq k$ (in particular, k claims s entails s true). This entailment does not hold in general.

The rules of our sequent calculus are summarized in Figure 2. As usual, we have left and right rules for each connective. For common connectives, the rules resemble those in intuitionistic logic, with the exception of the associated context, which remains unchanged. The judgment $\Sigma \vdash t{:}\sigma$ means that the term t has sort σ. We restrict the (init) rule to atomic formulas only. This is merely a technical convenience because we prove later that $\Sigma; \Gamma, s \xrightarrow{k} s$ for arbitrary s. Rule (claims) enforces the meaning of contexts as described above by allowing k claims s on the left to be promoted to s if the context k_0 is weaker than k.

(saysR) is the only rule which changes the context of a sequent. The notation $\Gamma|$ in this rule denotes the subset of Γ that contains exactly the claims of principals, i.e., the set $\{(k' \text{ claims } s') \in \Gamma\}$. The rule means that k says s is true in any context k_0 if s is true in context k using only claims of principals. Truth assumptions in Γ are eliminated in the premise because they may have been added in the context k_0 (using the rules (claims) and ($\supset$R)), but may not hold in the context k. The left rule (saysL) changes the judgment (k says s) true to the equivalent judgment k claims s.

Meta-theory. We prove two important meta-theorems about this sequent calculus: admissibility of the cut rule and the identity principle. In addition, common structural theo-

[1] Often in literature, context is used to refer to Γ. However, we consistently use context for k and hypotheses for Γ.

$$\frac{}{\Sigma;\Gamma,p \xrightarrow{k} p}\text{init} \qquad \frac{k \succeq k_0 \qquad \Sigma;\Gamma,k \text{ claims } s, s \xrightarrow{k_0} r}{\Sigma;\Gamma,k \text{ claims } s \xrightarrow{k_0} r}\text{claims} \qquad \frac{\Sigma;\Gamma| \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k_0} k \text{ says } s}\text{saysR}$$

$$\frac{\Sigma;\Gamma,k \text{ says } s, k \text{ claims } s \xrightarrow{k_0} r}{\Sigma;\Gamma,k \text{ says } s \xrightarrow{k_0} r}\text{saysL} \qquad \frac{\Sigma;\Gamma \xrightarrow{k} s \qquad \Sigma;\Gamma \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \wedge s'}\wedge\text{R} \qquad \frac{\Sigma;\Gamma,s \wedge s', s, s' \xrightarrow{k} r}{\Sigma;\Gamma,s \wedge s' \xrightarrow{k} r}\wedge\text{L}$$

$$\frac{\Sigma;\Gamma \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k} s \vee s'}\vee\text{R}_1 \qquad \frac{\Sigma;\Gamma \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \vee s'}\vee\text{R}_2 \qquad \frac{\Sigma;\Gamma,s \vee s', s \xrightarrow{k} r \qquad \Sigma;\Gamma,s \vee s', s' \xrightarrow{k} r}{\Sigma;\Gamma,s \vee s' \xrightarrow{k} r}\vee\text{L}$$

$$\frac{}{\Sigma;\Gamma \xrightarrow{k} \top}\top\text{R} \qquad \frac{}{\Sigma;\Gamma,\bot \xrightarrow{k} r}\bot\text{L} \qquad \frac{\Sigma;\Gamma,s \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \supset s'}\supset\text{R}$$

$$\frac{\Sigma;\Gamma,s \supset s' \xrightarrow{k} s \qquad \Sigma;\Gamma,s \supset s', s' \xrightarrow{k} r}{\Sigma;\Gamma,s \supset s' \xrightarrow{k} r}\supset\text{L} \qquad \frac{\Sigma,x{:}\sigma;\Gamma \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k} \forall x{:}\sigma.s}\forall\text{R}$$

$$\frac{\Sigma;\Gamma,\forall x{:}\sigma.s, s[t/x] \xrightarrow{k} r \qquad \Sigma \vdash t{:}\sigma}{\Sigma;\Gamma,\forall x{:}\sigma.s \xrightarrow{k} r}\forall\text{L} \qquad \frac{\Sigma;\Gamma \xrightarrow{k} s[t/x] \qquad \Sigma \vdash t{:}\sigma}{\Sigma;\Gamma \xrightarrow{k} \exists x{:}\sigma.s}\exists\text{R}$$

$$\frac{\Sigma,x{:}\sigma;\Gamma,\exists x{:}\sigma.s, s \xrightarrow{k} r}{\Sigma;\Gamma,\exists x{:}\sigma.s \xrightarrow{k} r}\exists\text{L}$$

Figure 2: BL_0: sequent calculus

rems such as weakening and strengthening of hypotheses are also provable, but we do not state them explicitly.

Theorem 2.1 (Identity)**.** $\Sigma;\Gamma,s \xrightarrow{k} s$ *for each* s*.*

Proof. By induction on s. □

Theorem 2.2 (Admissibility of cut)**.** *The following hold*

1. $\Sigma;\Gamma \xrightarrow{k} s$ *and* $\Sigma;\Gamma,s \xrightarrow{k} r$ *imply* $\Sigma;\Gamma \xrightarrow{k} r$
2. $\Sigma;\Gamma| \xrightarrow{k} s$ *and* $\Sigma;\Gamma,k \text{ claims } s \xrightarrow{k_0} r$ *imply* $\Sigma;\Gamma \xrightarrow{k_0} r$

Proof. By simultaneous lexicographic induction, first on the size of the cut formula, and then on the sizes of the two given derivations, as in [45]. □

Finally, we prove an equivalence between the axiomatic system and the sequent calculus.

Theorem 2.3 (Equivalence)**.** $\Sigma; \cdot \xrightarrow{k} s$ *in the sequent calculus if and only if* $\vdash k$ says s *in the axiomatic system.*

Proof. The "if" direction follows by a direct induction on axiomatic proofs. For the "only if" direction, we generalize the statement of the theorem to allow non-empty hypothesis: if $\Sigma; \Gamma \xrightarrow{k} s$, then $\vdash k$ says $(\Gamma \supset s)$. Next, we generalize the axiomatic system to allow hypothesis. Finally, we induct on sequent derivations to show that they can be simulated in the generalized axiomatic system. Although tedious, this approach is standard. □

Using the sequent calculus we can prove that BL_0 is consistent ($\bot$ cannot be derived), and that k says $(s \wedge r)$ is provably equivalent to $(k \text{ says } s) \wedge (k \text{ says } r)$. Similar distributive properties do not hold for other connectives. Also, neither s nor k says s implies the other in general.

3 Goal-directed Search in BL_0

We now turn to the problem of proving BL_0 sequents through logic programming. Our broad objective is to identify fragments of the logic on which we can describe simple, complete search strategies adhering to two common paradigms: (1) backward chaining or goal-directed search, the method used in Prolog, and (2) forward chaining or saturating search, the approach used in Datalog. Goal-directed search is covered in this section, whereas saturating search is described in the next section. Before proceeding with technical details, we make three important observations.

First, both goal-directed and saturating search can be seen as instances of a general technique called *focusing* [7]. Although we may apply this technique to the whole logic, and use it construct a generic theorem prover for BL_0, we refrain from doing so here. Such a generic tool would be needlessly complicated and possibly slow because it would have to cater to many cases that may never arise in actual policies.

Second, the distinction between goal-directed search and saturating search should not be confused with the distinction between backwards and forwards application of inference rules (the latter is also known as the inverse method). Both techniques described here apply inference rules backwards, i.e., a sequent to be proved is matched against the conclusion of a rule, whose premises are then recursively established. The difference between goal-directed search and saturating search arises because the rules used in the two cases are different.

Third, for some applications such as proof-carrying authorization [8, 9] and also sometimes for keeping logs of access, it is essential to construct and output proof-terms. Adding proof-terms to the logic programming strategies described below is relatively straightforward, and even though our implementation produces them, we do not describe them here.

Goal-directed search. The term *goal* refers to the formula on the right side of a sequent to be proved. Goal-directed search, in the sense that we use it here, is characterized by two principles: (1) Unless a goal is atomic, it is immediately decomposed using the right rule of its top most connective. Proofs constructed in this manner have previously been called

uniform proofs [43]. (2) When the goal is atomic, a hypothesis that can potentially prove the goal is chosen (technically, the *head* of the hypothesis must match the goal), decomposed completely using left rules, producing several additional goals to be proved, which are then established recursively. This second principle is called backward chaining.

Although this search strategy is incomplete for the whole logic (for e.g., it cannot establish $\Sigma; s \vee r \xrightarrow{k} s \vee r$), there are fragments of the logic on which it is complete. We describe one such fragment in detail below. This fragment extends the so called Hereditary-Harrop (HH) fragment of first order logic, which is known to admit a complete goal-directed search strategy [43]. We show that its extension with says also admits a similar strategy. Although non-trivial, this is a modular extension of the first-order case.

Hereditary-Harrop fragment of $\mathbf{BL}_0$. The syntax of the Hereditary-Harrop (HH) fragment of BL_0 is described below. Goals g may contain all connectives. Hypotheses are divided into two categories: clauses d that contain only *negative connectives* ($\supset, \wedge, \top, \forall$), and chunks h that contain *positive connectives* ($\mathsf{says}, \wedge, \vee, \top, \bot, \exists$). The left sequent calculus rules of positive connectives are invertible, i.e., they can be eagerly applied in the backwards direction without retaining the principal formula in the premise or losing completeness. The left rules of negative connectives are not invertible, and they are applied only when the result of the application will be useful. ($\wedge$ is a special connective – its left rules can be written in two different ways, one of which is invertible, and the other of which is not. Hence, it is both positive and negative. Figure 2 shows only the invertible rule.)

$$
\begin{array}{llll}
\text{Goals} & g & ::= & p \mid g_1 \wedge g_2 \mid g_1 \vee g_2 \mid h \supset g \mid \top \mid \bot \mid \forall x{:}\sigma.g \mid \exists x{:}\sigma.g \mid k \ \mathsf{says}\ g \\
\text{Clauses} & d & ::= & p \mid g \supset d \mid d_1 \wedge d_2 \mid \top \mid \forall x{:}\sigma.d \\
\text{Chunks} & h & ::= & d \mid k \ \mathsf{says}\ d \mid h_1 \wedge h_2 \mid h_1 \vee h_2 \mid \top \mid \bot \mid \exists x{:}\sigma.h \\
\text{Policies} & \Delta & ::= & \cdot \mid \Delta, d \mid \Delta, k \ \mathsf{claims}\ d \\
\text{Groups} & \Xi & ::= & \cdot \mid \Xi, h \\
\text{Sequents} & \mathrm{R} & ::= & \Sigma; \Delta \xRightarrow{k} g \\
 & \mathrm{L} & ::= & \Sigma; \Delta; \Xi \xLeftarrow{k} g \\
 & \mathrm{N} & ::= & \Sigma; \Delta \xLeftrightarrow{k} p \\
 & \mathrm{F} & ::= & \Sigma; d \ll p \mid g_1 \ldots g_n
\end{array}
$$

Hypotheses may be of two kinds. *Policies* Δ contain assumptions of the the forms d or k claims d, whereas *groups* Ξ contain chunks. We need four kinds of sequents (labeled R,L,N,F), distinguished by the portions in which rules are applied. In R sequents $\Sigma; \Delta \xRightarrow{k} g$, the goal is decomposed by right rules. This corresponds to principle (1) above. In L sequents $\Sigma; \Delta; \Xi \xLeftarrow{k} g$, the chunks in Ξ are decomposed using left rules. In N sequents $\Sigma; \Delta \xLeftrightarrow{k} p$, which cause backward chaining, we choose a clause from the hypotheses that can be decomposed to prove the atomic goal p, decompose the clause, and prove the resulting subgoals. This corresponds to principle (2) above. F sequents $\Sigma; d \ll p \mid g_1 \ldots g_n$ decompose the clause d, proving atomic goal p, which is an input and generate subgoals $g_1 \ldots g_n$, which are outputs.

R sequents

$$\frac{\Sigma;\Delta \overset{k}{\Leftrightarrow} p}{\Sigma;\Delta \overset{k}{\Rightarrow} p}\text{atom} \qquad \frac{\Sigma;\Delta| \overset{k}{\Rightarrow} g}{\Sigma;\Delta \overset{k_0}{\Rightarrow} k \text{ says } g}\mathsf{saysR} \qquad \frac{\Sigma;\Delta \overset{k}{\Rightarrow} g \qquad \Sigma;\Delta \overset{k}{\Rightarrow} g'}{\Sigma;\Delta \overset{k}{\Rightarrow} g \wedge g'}\wedge\text{R}$$

$$\frac{\Sigma;\Delta \overset{k}{\Rightarrow} g}{\Sigma;\Delta \overset{k}{\Rightarrow} g \vee g'}\vee\text{R}_1 \qquad \frac{\Sigma;\Delta \overset{k}{\Rightarrow} g'}{\Sigma;\Delta \overset{k}{\Rightarrow} g \vee g'}\vee\text{R}_2 \qquad \frac{}{\Sigma;\Delta \overset{k}{\Rightarrow} \top}\top\text{R} \qquad \frac{\Sigma;\Delta;h \overset{k}{\Leftarrow} g'}{\Sigma;\Delta \overset{k}{\Rightarrow} h \supset g'}\supset\text{R}$$

$$\frac{\Sigma, x{:}\sigma;\Delta \overset{k}{\Rightarrow} g}{\Sigma;\Delta \overset{k}{\Rightarrow} \forall x{:}\sigma.g}\forall\text{R} \qquad \frac{\Sigma;\Delta \overset{k}{\Rightarrow} g[t/x] \qquad \Sigma \vdash t{:}\sigma}{\Sigma;\Delta \overset{k}{\Rightarrow} \exists x{:}\sigma.g}\exists\text{R}$$

L sequents

$$\frac{\Sigma;\Delta \overset{k}{\Rightarrow} g}{\Sigma;\Delta;\cdot \overset{k}{\Leftarrow} g}\Rightarrow\Leftarrow \qquad \frac{\Sigma;\Delta,d;\Xi \overset{k}{\Leftarrow} g}{\Sigma;\Delta;\Xi,d \overset{k}{\Leftarrow} g}\text{pr} \qquad \frac{\Sigma;\Delta,k \text{ claims } d;\Xi \overset{k_0}{\Leftarrow} g}{\Sigma;\Delta;\Xi,k \text{ says } d \overset{k_0}{\Leftarrow} g}\mathsf{saysL}$$

$$\frac{\Sigma;\Delta;\Xi,h,h' \overset{k}{\Leftarrow} g}{\Sigma;\Delta;\Xi,h \wedge h' \overset{k}{\Leftarrow} g}\wedge\text{L} \qquad \frac{\Sigma;\Delta;\Xi,h \overset{k}{\Leftarrow} g \qquad \Sigma;\Delta;\Xi,h' \overset{k}{\Leftarrow} g}{\Sigma;\Delta;\Xi,h \vee h' \overset{k}{\Leftarrow} g}\vee\text{L} \qquad \frac{\Sigma;\Delta;\Xi \overset{k}{\Leftarrow} g}{\Sigma;\Delta;\Xi,\top \overset{k}{\Leftarrow} g}\top\text{L}$$

$$\frac{}{\Sigma;\Delta;\Xi,\bot \overset{k}{\Leftarrow} g}\bot\text{L} \qquad \frac{\Sigma, x{:}\sigma;\Delta;\Xi,h \overset{k}{\Leftarrow} g}{\Sigma;\Delta;\Xi,\exists x{:}\sigma.h \overset{k}{\Leftarrow} g}\exists\text{L}$$

F sequents

$$\frac{}{\Sigma;p \ll p \mid \cdot}\text{init} \qquad \frac{\Sigma;d_1 \ll p \mid G}{\Sigma;d_1 \wedge d_2 \ll p \mid G}\wedge\text{L}_1 \qquad \frac{\Sigma;d_2 \ll p \mid G}{\Sigma;d_1 \wedge d_2 \ll p \mid G}\wedge\text{L}_2$$

$$\frac{\Sigma;d_2 \ll p \mid G}{\Sigma;g_1 \supset d_2 \ll p \mid g_1, G}\supset\text{L} \qquad \frac{\Sigma;d[t/x] \ll p \mid G \qquad \Sigma \vdash t{:}\sigma}{\Sigma;\forall x{:}\sigma.d \ll p \mid G}\forall\text{L}$$

N sequents

$$\frac{d \in \Delta \qquad \Sigma;d \ll p \mid g_1 \ldots g_n \qquad (\Sigma;\Delta \overset{k}{\Rightarrow} g_i)_{i=1}^n}{\Sigma;\Delta \overset{k}{\Leftrightarrow} p}\text{B1}$$

$$\frac{k \text{ claims } d \in \Delta \qquad k \succeq k_0 \qquad \Sigma;d \ll p \mid g_1 \ldots g_n \qquad (\Sigma;\Delta \overset{k_0}{\Rightarrow} g_i)_{i=1}^n}{\Sigma;\Delta \overset{k_0}{\Leftrightarrow} p}\text{B2}$$

Figure 3: HH: Goal-directed search (contd.)

The rules for goal-directed search are shown in Figure 3. All rules are applied backwards. Search starts in an R sequent (it may start in an L sequent if the hypotheses contain chunks). In this form, right rules of the top-level connective of g are successively applied to decompose the goal to atomic formulas. An atomic goal transitions to an N sequent (rule (atom)). The only exception to this is the rule ($\supset$R), which introduces a chunk on the left and transitions temporarily to an L sequent.

In an L sequent, chunks in Ξ are decomposed using left rules of their top level connectives. The rules may be applied to chunks in any order. Usually, a canonical left-to-right or right-to-left order is chosen. Chunks eventually decompose into assumptions of form d or k says d, which are pushed into the policies Δ in the forms d and k claims d respectively (rules (pr) and (saysL)). Eventually Ξ becomes empty, there is a transition to an R sequent (rule ($\Rightarrow\Leftarrow$)), and decomposition of the goal is resumed.

An N sequent can be established in essentially one way: find a suitable clause to prove its atomic goal, decompose the clause using left rules of its connectives successively (judgment $\Sigma; d \ll p \mid g_1 \dots g_n$), and solve the generated subgoals $g_1 \dots g_n$. There are two rules to prove an N sequent, one where the clause d is directly picked from Δ (B1), and another where the clause d is picked from an assumption k claims d (B2). In the latter case, k must be stronger than the context (else the assumption k claims d does not entail d and is hence not usable).

An F sequent is given as input a clause d and a formula p. It is provable with output $g_1 \dots g_n$ if using left rules, d can be decomposed, revealing p at its head. In general, if $\Sigma; d \ll p \mid g_1 \dots g_n$, $d \in \Delta$, and $\Sigma; \Delta \xrightarrow{k} g_i$ for each i, then $\Sigma; \Delta \xrightarrow{k} p$. A proof of an F sequent corresponds to a single step of *left focusing*.

Example 2. We revisit Example 1 and show some of the steps in constructing a proof of access using goal-directed search. Let us call the set of policies (1)–(10) from the earlier example Ξ and let Σ define the constants used in the policy. We start goal-directed search with the L sequent $\Sigma; \cdot; \Xi \overset{k_0}{\Leftarrow} \mathsf{admin}\ \mathsf{says}\ \mathtt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$, where k_0 is an arbitrary principal (we may choose $k_0 = \ell$). Using the rules (saysL) and ($\Rightarrow\Leftarrow$), this reduces to proving $\Sigma; \Delta \overset{k_0}{\Rightarrow} \mathsf{admin}\ \mathsf{says}\ \mathtt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$, where Δ is the same as Ξ, except that all top-level says are replaced by claims. Since this is an R sequent, we must use the rules (saysR) and (atom) in order to get $\Sigma; \Delta \overset{\mathsf{admin}}{\Leftrightarrow} \mathtt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$ (note that $\Delta| = \Delta$). Now we have an N sequent, hence a hypothesis must be picked. The only clause that will successfully prove the goal is the one contained in policy (1). We call this clause $\mathtt{one}$ here. Indeed we can show, $\Sigma; \mathtt{one} \ll \mathtt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt}) \mid g$, where

$$\begin{aligned} g \;=\; & \mathsf{hr}\ \mathsf{says}\ \mathtt{employee}(\mathsf{Bob}) \wedge \mathtt{hasLevelForFile}(\mathsf{Bob}, \mathsf{secret.txt}) \wedge \\ & \mathsf{system}\ \mathsf{says}\ \mathtt{owns}(\mathsf{Alice}, \mathsf{secret.txt}) \wedge \mathsf{Alice}\ \mathsf{says}\ \mathtt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt}) \end{aligned}$$

Hence the rule (B2) may be used, generating the following premise $\Sigma; \Delta \overset{\mathsf{admin}}{\Rightarrow} g$. Only the rule ($\wedge$R) applies to this R sequent, resulting in the following premises:

$$\Sigma; \Delta \overset{\mathsf{admin}}{\Rightarrow} \mathsf{hr\ says\ employee(Bob)}$$
$$\Sigma; \Delta \overset{\mathsf{admin}}{\Rightarrow} \mathtt{hasLevelForFile}(\mathsf{Bob}, \mathsf{secret.txt})$$
$$\Sigma; \Delta \overset{\mathsf{admin}}{\Rightarrow} \mathsf{system\ says\ owns(Alice, secret.txt)}$$
$$\Sigma; \Delta \overset{\mathsf{admin}}{\Rightarrow} \mathsf{Alice\ says\ may(read, Bob, secret.txt)}$$

We show further only how the first of these premises is established. Using the rules (saysR) and (atom), we obtain $\Sigma; \Delta \overset{\mathsf{hr}}{\Leftrightarrow} \mathtt{employee}(\mathsf{Bob})$. Now we may use the clause in policy (8) (call it `eight`). Since $\Sigma; \mathtt{eight} \ll \mathtt{employee}(\mathsf{Bob}) \mid \cdot$, use of (B2) generates no new premises, and the proof branch closes.

Observe that as we construct the proof backwards, the context changes, e.g., from k_0 to admin to hr above. The change of context makes new assumptions in the hypotheses usable via the rule (B2). For instance, we could not have used policy (8) when the context was admin because it is not the case that $\mathsf{hr} \succeq \mathsf{admin}$. This has practical implications. Since the claims of only some of the principals are usable at any time (precisely, those principals that are stronger than the context), a prover may choose to load the hypotheses lazily. In the above illustration, the prover may choose not to load the policies (1) and (2) until the context is admin, and may not load policy (8) until the context is hr, without effecting completeness. This may be very useful if policies are distributed on remote servers. In that case, policies may be "fetched" on demand depending on the current context.

Eliminating non-determinism. The proof system of Figure 3 is largely deterministic. The only points of non-determinism are: (a) Choosing a clause in rules (B1) and (B2). (b) Choosing between rules $\vee\mathrm{R}_1$ and $\vee\mathrm{R}_2$. (c) Choosing between rules $\wedge\mathrm{L}_1$ and $\wedge\mathrm{L}_2$. (d) Finding appropriate terms t in rules $\exists$R and $\forall$L. Of these, (a), (b), and (c) cause backtracking during search, which can be reduced significantly in practice through clause compilation and term-indexing (see e.g., [50]). (d) can be eliminated completely through unification. All these techniques are standard in logic programming and apply to BL_0 without modification, so we do not discuss them further.

Soundness and Completeness. It is quite easy to prove by induction on the rules of Figure 3 that goal-directed search is sound, i.e., any proof it constructs can be simulated in the sequent calculus. The converse, completeness, is harder to prove and requires a number of properties about the sequent calculus. We state our correctness theorem below, postponing an outline of lemmas needed for completeness to Appendix A.

Theorem 3.1 (Correctness of goal-directed search)**.**

1. *Soundness: If* $\Sigma; \Delta \overset{k}{\Rightarrow} g$*, then* $\Sigma; \Delta \overset{k}{\rightarrow} g$
2. *Completeness: If* $\Sigma; \Delta \overset{k}{\rightarrow} g$*, then* $\Sigma; \Delta \overset{k}{\Rightarrow} g$*.*

4 Saturating Search in $\mathbf{BL}_0$

Saturating search is the method of proof construction used in Datalog. The basic tenet is to decompose clauses eagerly by left rules, adding new assumptions to the hypotheses, without any regard for what may or may not be useful in proving the goal. This process of adding assumptions is called forward chaining. After enough assumptions have been added (or when no more can be added), the goal is decomposed. Search succeeds if each atomic formula produced by decomposing the goal exists in the hypotheses else it fails. This is in stark contrast with goal-directed search, where a clause is decomposed only if its head matches an atomic goal. Saturating search is useful in many cases, for example when all possible consequences of a policy are needed.

Like goal-directed search, saturating search is incomplete for the whole logic. For example, it cannot be used to prove $\Sigma; \forall x{:}\sigma.P(x) \xrightarrow{k} \forall x{:}\sigma.P(x)$. Once again, we identify a fragment of BL_0 on which saturating search is complete. For the lack of a better name, we call this fragment the forward-Hereditary-Harrop fragment (FHH). In saturating search the entire hypotheses must be available at once; the lazy strategy for loading hypothesis on demand that we discussed for goal-directed search in Example 2 does not work here.

Forward-Hereditary-Harrop Fragment of $\mathbf{BL}_0$. The syntax of FHH is shown below. Goals contain only positive connectives, and the heads of clauses are chunks, not atomic formulas. Another difference from HH is that, in the hypotheses, says is restricted to atomic formulas and all atomic formulas must be enclosed by an immediate says. These restrictions are necessary to obtain completeness.

$$
\begin{array}{llll}
\text{Goals} & g & ::= & p \mid k \ \mathsf{says}\ g \mid g_1 \wedge g_2 \mid g_1 \vee g_2 \mid \top \mid \bot \mid \exists x{:}\sigma.g \\
\text{Clauses} & d & ::= & h \mid g \supset d \mid d_1 \wedge d_2 \mid \top \mid \forall x{:}\sigma.d \\
\text{Chunks} & h & ::= & k \ \mathsf{says}\ p \mid h_1 \wedge h_2 \mid h_1 \vee h_2 \mid \top \mid \bot \mid \exists x{:}\sigma.h \\
\text{Policies} & \Delta & ::= & \cdot \mid \Delta, d \mid \Delta, k \ \mathsf{claims}\ p \\
\text{Groups} & \Xi & ::= & \cdot \mid \Xi, h \\
\text{Sequents} & \mathrm{R} & ::= & \Sigma; \Delta \overset{k}{\gg} g \\
 & \mathrm{L} & ::= & \Sigma; \Delta; \Xi \overset{k}{\Leftarrow} g \\
 & \mathrm{N} & ::= & \Sigma; \Delta \overset{k}{\Leftrightarrow} g \\
 & \mathrm{F} & ::= & \Sigma; d \ll h \mid g_1 \ldots g_n
\end{array}
$$

As before, we use four types of sequents R,L,N,F. However, R sequents which played a central role in goal-directed search have a secondary role here and occur only at the leaves of the derivation. Instead N sequents are central. The context k changes only in R sequents.

Figure 4 summarizes the rules for saturating search. Search starts with an N sequent, where all forward chaining happens (search may start with an L sequent if the hypotheses contain chunks). There are two ways to prove an N sequent. Either we may end forward chaining, and transition to an R sequent which would decompose the goal (rule (F1)), or we may continue forward chaining (rule (F2)). In the latter case, we pick a clause d from the hypotheses, decompose it using F sequents (premise $\Sigma; d \ll h \mid g_1 \ldots g_n$), solve the goals

R sequents

$$\frac{k' \;\mathsf{claims}\; p \in \Delta \qquad k' \succeq k}{\Sigma; \Delta \overset{k}{\gg} p}\text{init} \qquad \frac{\Sigma; \Delta| \overset{k}{\gg} g}{\Sigma; \Delta \overset{k'}{\gg} k \;\mathsf{says}\; g}\mathsf{saysR} \qquad \frac{\Sigma; \Delta \overset{k}{\gg} g \qquad \Sigma; \Delta \overset{k}{\gg} g'}{\Sigma; \Delta \overset{k}{\gg} g \wedge g'}\wedge\mathrm{R}$$

$$\frac{\Sigma; \Delta \overset{k}{\gg} g}{\Sigma; \Delta \overset{k}{\gg} g \vee g'}\vee\,\mathrm{R}_1 \qquad \frac{\Sigma; \Delta \overset{k}{\gg} g'}{\Sigma; \Delta \overset{k}{\gg} g \vee g'}\vee\,\mathrm{R}_2 \qquad \frac{}{\Sigma; \Delta \overset{k}{\gg} \top}\top\mathrm{R} \qquad \frac{\Sigma; \Delta \overset{k}{\gg} g[t/x] \qquad \Sigma \vdash t{:}\sigma}{\Sigma; \Delta \overset{k}{\gg} \exists x{:}\sigma.g}\exists\mathrm{R}$$

L sequents

$$\frac{\Sigma; \Delta \overset{k}{\Leftrightarrow} g}{\Sigma; \Delta; \cdot \overset{k}{\Leftarrow} g}\Leftrightarrow\Leftarrow \qquad \frac{\Sigma; \Delta, k' \;\mathsf{claims}\; p; \Xi \overset{k}{\Leftarrow} g}{\Sigma; \Delta; \Xi, k' \;\mathsf{says}\; p \overset{k}{\Leftarrow} g}\mathsf{saysL} \qquad \frac{\Sigma; \Delta; \Xi, h, h' \overset{k}{\Leftarrow} g}{\Sigma; \Delta; \Xi, h \wedge h' \overset{k}{\Leftarrow} g}\wedge\mathrm{L}$$

$$\frac{\Sigma; \Delta; \Xi \overset{k}{\Leftarrow} g}{\Sigma; \Delta; \Xi, \top \overset{k}{\Leftarrow} g}\top\mathrm{L} \qquad \frac{\Sigma; \Delta; \Xi, h \overset{k}{\Leftarrow} g \qquad \Sigma; \Delta; \Xi, h' \overset{k}{\Leftarrow} g}{\Sigma; \Delta; \Xi, h \vee h' \overset{k}{\Leftarrow} g}\vee\,\mathrm{L} \qquad \frac{}{\Sigma; \Delta; \Xi, \bot \overset{k}{\Leftarrow} g}\bot\mathrm{L}$$

$$\frac{\Sigma, x{:}\sigma; \Delta; \Xi, h \overset{k}{\Leftarrow} g}{\Sigma; \Delta; \Xi, \exists x{:}\sigma.h \overset{k}{\Leftarrow} g}\exists\mathrm{L}$$

F sequents

$$\frac{}{\Sigma; h \ll h \mid \cdot}\text{blur} \qquad \frac{\Sigma; d_1 \ll h \mid G}{\Sigma; d_1 \wedge d_2 \ll h \mid G}\wedge\mathrm{L}_1 \qquad \frac{\Sigma; d_2 \ll h \mid G}{\Sigma; d_1 \wedge d_2 \ll h \mid G}\wedge\mathrm{L}_2$$

$$\frac{\Sigma; d_2 \ll h \mid G}{\Sigma; g_1 \supset d_2 \ll h \mid g_1, G}\supset\mathrm{L} \qquad \frac{\Sigma; d[t/x] \ll h \mid G \qquad \Sigma \vdash t{:}\sigma}{\Sigma; \forall x{:}\sigma.d \ll h \mid G}\forall\mathrm{L}$$

N sequents

$$\frac{\Sigma; \Delta \overset{k}{\gg} g}{\Sigma; \Delta \overset{k}{\Leftrightarrow} g}\mathrm{F1} \qquad \frac{d \in \Delta \qquad \Sigma; d \ll h \mid g_1 \ldots g_n \qquad (\Sigma; \Delta \overset{k}{\gg} g_i)_{i=1}^{n} \qquad \Sigma; \Delta; h \overset{k}{\Leftarrow} g}{\Sigma; \Delta \overset{k}{\Leftrightarrow} g}\mathrm{F2}$$

Figure 4: FHH: Saturating search

$g_1 \ldots g_n$, and then decompose the chunk h in an L sequent. This has the effect of adding all components of h to the hypotheses, and constitutes a single forward chaining step.

In an R sequent, we decompose a goal using the right rule of its top-level connective, till we reach an atomic goal. At this point we simply check if k' claims p exists in the hypotheses for some $k' \succeq k$ (rule (init)). If this happens to be the case, the branch closes, else the branch fails. As a result, all assumptions needed to prove the leaves of the goal must be directly present in the hypotheses before the goal is decomposed.

An L sequent is reached from an N sequent (rule (B2), fourth premise). It decomposes chunks in Ξ using left rules. After Ξ becomes empty, it transitions back to an N sequent (rule $\Leftrightarrow\Leftarrow$), and forward chaining resumes. This is slightly different from goal-directed search where L sequents arise from R sequents and lead back to them.

As in goal-directed search F sequents in FHH decompose clauses. However, in place of taking as *input* an atomic formula to prove, they produce as *output* a chunk. The subgoals $g_1 \ldots g_n$ are outputs as before.

Example 3. Returning to our running example, we observe that the syntax used in policies (1) and (2) is outside the FHH fragment. Now we rewrite these policies in FHH. The consequences of these new policies (1f and 2f below) are the same as those of the original ones. We formalize this equivalence in Section 5.

$$\text{(1f)}\quad \begin{array}{l}\forall k, k', f.\ (((\mathsf{hr\ says\ employee}(k)) \wedge (\mathsf{admin\ says\ hasLevelForFile}(k, f)) \wedge \\ \qquad (\mathsf{system\ says\ owns}(k', f)) \wedge (k'\ \mathsf{says\ may}(\mathsf{read}, k, f))) \\ \qquad \supset \mathsf{admin\ says\ may}(\mathsf{read}, k, f))\end{array}$$

$$\text{(2f)}\quad \begin{array}{l}\forall k, f, l, l'.\ (((\mathsf{system\ says\ levelFile}(f, l)) \wedge (\mathsf{hr\ says\ levelPrin}(k, l')) \wedge \\ \qquad (\mathsf{admin\ says\ below}(l, l'))) \supset \mathsf{admin\ says\ hasLevelForFile}(k, f))\end{array}$$

The policies (1f), (2f), and (3)–(10) lie in FHH. Let us call the set of these policies Ξ. As in Example 2, we show some of the steps in proving an access from these policies. At the top level we want to prove $\Sigma; \cdot; \Xi \overset{k_0}{\Leftarrow} \mathsf{admin\ says\ may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$, where k_0 is a fresh constant. Using the rules (saysL) and $(\Leftrightarrow\Leftarrow)$, this reduces to proving $\Sigma; \Delta \overset{k_0}{\Leftrightarrow} \mathsf{admin\ says\ may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$, where Δ is the same as Ξ, except that all top-level says in policies are replaced by claims. Since this is an N sequent, we may now start forward chaining. The cardinal rule of forward chaining is that a clause can be decomposed only if the subgoals that it will generate are directly provable through R sequents. (1f) does not satisfy this condition, but (2f) does. Accordingly, we decompose it using an F sequent. It is easy to see that $\Sigma; \text{2f} \ll \mathsf{admin\ says\ hasLevelForFile}(\mathsf{Bob}, \mathsf{secret.txt}) \mid g$, where

$$\begin{array}{rcl} g & = & \mathsf{system\ says\ levelFile}(\mathsf{secret.txt}, \mathsf{secret}) \wedge \\ & & \mathsf{hr\ says\ levelPrin}(\mathsf{Bob}, \mathsf{topsecret}) \wedge \\ & & \mathsf{admin\ says\ below}(\mathsf{secret}, \mathsf{topsecret}) \end{array}$$

Thus application of (F2) produces the new sequent $\Sigma; \Delta \overset{k_0}{\gg} g$ in the third premise. Only the ($\wedge$R) rule applies to this R sequent, resulting in the following premises: ($\Sigma; \Delta \overset{k_0}{\gg}$

system says $\texttt{levelFile}(\mathsf{secret.txt}, \mathsf{secret})$), $(\Sigma; \Delta \overset{k_0}{\gg} \mathsf{hr}$ says $\texttt{levelPrin}(\mathsf{Bob}, \mathsf{topsecret}))$, and $(\Sigma; \Delta \overset{k_0}{\gg} \mathsf{admin}$ says $\texttt{below}(\mathsf{secret}, \mathsf{topsecret}))$. All these sequents close immediately by the rules (saysR) and (init). Hence in the last premise of (F2) we get $\Sigma; \Delta; \mathsf{admin}$ says $\texttt{hasLevelForFile}(\mathsf{Bob}, \mathsf{secret.txt}) \overset{k_0}{\Leftarrow} \mathsf{admin}$ says $\texttt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$. The rules (saysL) and $(\Leftrightarrow\Leftarrow)$ must now be applied in sequence to obtain $\Sigma; \Delta, \mathsf{admin}$ claims $\texttt{hasLevelForFile}$ $(\mathsf{Bob}, \mathsf{secret.txt}) \overset{k_0}{\Leftrightarrow} \mathsf{admin}$ says $\texttt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$. This is an N sequent, and we may choose to use (F2) again, decomposing (1f) and introducing the assumption admin claims $\texttt{may}(\mathsf{read}, \mathsf{Bob}, \mathsf{secret.txt})$ in a similar manner. The rest of the proof is easily constructed: (F1) is used to shift to an R sequent, which closes immediately by rules (saysR) and (init).

A salient point to observe here is that the clauses decomposed in rule (F2) need to be guessed. Further, there is no heuristic to determine when (F1) should be used instead of (F2). In general, a reasonable strategy for a saturating prover is to keep applying (F2) until no new hypotheses can be generated. At that point (F1) may be used to try to close the proof. When the objective is to find all consequences of a hypotheses, a goal is not provided and (F1) is never used. Instead, the saturated Δ is the output of the search. Of course, this may result in non-termination if saturation never occurs.

Soundness and Completeness. Saturating search on the FHH fragment is sound and complete with respect to BL_0's sequent calculus. Soundness follows by a straightforward induction on the rules in Figure 4. Completeness is quite difficult to prove. Our proof extends recent work by Pfenning and Simmons [48]. We state the correctness theorem below, deferring a description of the major steps in the proof of completeness to Appendix B. It is difficult to extend FHH and maintain completeness. For example, allowing goals containing $\supset$ or $\forall$, or not restricting says to atomic formulas in hypotheses results in incompleteness.

Theorem 4.1 (Correctness of saturating search)**.**

1. *Soundness: If* $\Sigma; \Delta \overset{k}{\Leftrightarrow} g$*, then* $\Sigma; \Delta \overset{k}{\rightarrow} g$
2. *Completeness: If* $\Sigma; \Delta \overset{k}{\rightarrow} g$*, then* $\Sigma; \Delta \overset{k}{\Leftrightarrow} g$*.*

5 From Goal-directed Search to Saturating Search and First-Order Logic

We consider a subfragment of HH, and show that it admits a sound and complete translation to FHH. Policies written in this fragment can therefore be used with both kinds of provers: they may be given directly to goal-directed provers for HH, and via the translation they may also be given to saturating provers for FHH. We go a step further and show that this smaller fragment also admits a sound and complete translation to intuitionistic first-order logic. Thus we may, in fact, use first-order provers on the fragment. The two translations are quite similar and we present them side by side. We call this smaller fragment of BL_0 Horn, because it looks similar to the Horn fragment of first-order logic.

	FHH			First-order		
Goals	$\langle P(t_1 \dots t_n)\rangle_k$	$=$	k says $P(t_1 \dots t_n)$	$[\![P(t_1 \dots t_n)]\!]_k$	$=$	$P(k, t_1 \dots t_n)$
	$\langle k' \text{ says } g\rangle_k$	$=$	$\langle g\rangle_{k'}$	$[\![k' \text{ says } g]\!]_k$	$=$	$[\![g]\!]_{k'}$
	$\langle g_1 \wedge g_2\rangle_k$	$=$	$\langle g_1\rangle_k \wedge \langle g_2\rangle_k$	$[\![g_1 \wedge g_2]\!]_k$	$=$	$[\![g_1]\!]_k \wedge [\![g_2]\!]_k$
	$\langle g_1 \vee g_2\rangle_k$	$=$	$\langle g_1\rangle_k \vee \langle g_2\rangle_k$	$[\![g_1 \vee g_2]\!]_k$	$=$	$[\![g_1]\!]_k \vee [\![g_2]\!]_k$
	$\langle \top\rangle_k$	$=$	$\top$	$[\![\top]\!]_k$	$=$	$\top$
	$\langle \bot\rangle_k$	$=$	k says $\bot$	$[\![\bot]\!]_k$	$=$	$\bot$
	$\langle \exists x{:}\sigma.g\rangle_k$	$=$	$\exists x{:}\sigma.\langle g\rangle_k$	$[\![\exists x{:}\sigma.g]\!]_k$	$=$	$\exists x{:}\sigma.[\![g]\!]_k$
Clauses	$\langle P(t_1 \dots t_n)\rangle_k$	$=$	k says $P(t_1 \dots t_n)$	$[\![P(t_1 \dots t_n)]\!]_k$	$=$	$P(k, t_1 \dots t_n)$
	$\langle \forall x{:}\sigma.d\rangle_k$	$=$	$\forall x{:}\sigma.\langle d\rangle_k$	$[\![\forall x{:}\sigma.d]\!]_k$	$=$	$\forall x{:}\sigma.[\![d]\!]_k$
	$\langle g \supset d\rangle_k$	$=$	$\langle g\rangle_k \supset \langle d\rangle_k$	$[\![g \supset d]\!]_k$	$=$	$[\![g]\!]_k \supset [\![d]\!]_k$
	$\langle \top\rangle_k$	$=$	$\top$	$[\![\top]\!]_k$	$=$	$\top$
	$\langle d_1 \wedge d_2\rangle_k$	$=$	$\langle d_1\rangle_k \wedge \langle d_2\rangle_k$	$[\![d_1 \wedge d_2]\!]_k$	$=$	$[\![d_1]\!]_k \wedge [\![d_2]\!]_k$
Policies	$\langle \cdot\rangle$	$=$	$\cdot$	$[\![\cdot]\!]$	$=$	$\cdot$
	$\langle \Delta, k \text{ claims } d\rangle$	$=$	$\langle \Delta\rangle, \langle d\rangle_k$	$[\![\Delta, k \text{ claims } d]\!]$	$=$	$[\![\Delta]\!], [\![d]\!]_k$

Figure 5: Translations from the Horn fragment to FHH and intuitionistic logic

Horn fragment of $\mathbf{BL}_0$. The syntax of the Horn fragment is shown below. We restrict goals to only positive connectives, while clauses contain only negative connectives. Hypotheses are always of the form k claims d. It is easy to check that this is a subfragment of HH and hence a complete goal-directed search strategy exists for it.

$$\begin{array}{llll} \text{Goals} & g & ::= & p \mid k \text{ says } g \mid g_1 \wedge g_2 \mid g_1 \vee g_2 \mid \top \mid \bot \mid \exists x{:}\sigma.g \\ \text{Clauses} & d & ::= & p \mid \forall x{:}\sigma.d \mid g \supset d \mid \top \mid d_1 \wedge d_2 \\ \text{Policies} & \Delta & ::= & \cdot \mid \Delta, k \text{ claims } d \end{array}$$

An important restriction in the Horn fragment is that we require that the order $\succeq$ be trivial, i.e., $k \succeq k'$ imply $k = k'$. In particular, ℓ is assumed to be absent. These restrictions are needed to prove correctness of the translation. Despite its simplicity, the Horn fragment is quite expressive. For instance, the example presented in this paper as well as the full policy from which it is derived can be expressed in this fragment through only very minor changes. It is our belief that this fragment is expressive enough to represent many policies of interest.

Translations. We assume that every predicate in BL_0 corresponds to a predicate of the same name in first-order logic that takes one extra argument. Figure 5 lists the translations $\langle\cdot\rangle$ and $[\![\cdot]\!]$ from the Horn fragment to FHH and intuitionistic first-order logic respectively. In both cases we use auxiliary translations that are indexed by a principal k. This k is the principal in the nearest says or claims outside the formula being translated.

The central "trick" of the translation to FHH is to push the says connective down to atomic formulas. That this simple idea works for a reasonably large fragment was surprising to the author. It was not obvious a priori that the nature of the says modality could be captured by pushing it to the leaves. We note, however, that the translation does not extend to significantly larger fragments. In particular, allowing either hypotheses of the form d true or other connectives in goals makes the translation unsound. The first-order translation is a mild extension of the FHH translation based on the observation that k says p behaves atomically in FHH (modulo the effects of the relation $k \succeq k'$, which we assume to be trivial in the Horn fragment). As a result, k says $P(t_1 \ldots t_n)$ can be replaced by $P(k, t_1 \ldots t_n)$.

Theorem 5.1 (Correctness of Translations)**.** *The following are equivalent for any policy* Δ *and any goal* g *in the Horn fragment.*

1. $\Sigma; \Delta \stackrel{k}{\Rightarrow} g$ *by the rules of HH.*
2. $\Sigma; \langle\Delta\rangle \stackrel{\ell}{\Leftrightarrow} \langle g\rangle_k$ *by the rules of FHH.*
3. $\Sigma; [\![\Delta]\!] \vdash [\![g]\!]_k$ *in first-order intuitionistic logic.*

Proof. Since we have already proved equivalence between HH (resp. FHH) and the sequent calculus, it suffices to prove that $\Sigma; \Delta \stackrel{k}{\rightarrow} g$ (resp. $\Sigma; \langle\Delta\rangle \stackrel{\ell}{\rightarrow} \langle g\rangle_k$) is equivalent to (3). In each case, and in each direction, this follows by a direct induction on given derivations. The inductions are easier if we subinduct on the size of g, and case analyze principal formulas in the last rules of the derivations. □

Connection to Binder and Soutei. The policy language Binder [23] contains a says modality that is extremely similar in nature to says in BL_0. In fact, Binder is almost a proper subset of the Horn fragment of BL_0, the only exception being that Binder allows says over clause heads. The semantics of Binder are *defined* by a translation to Datalog, that uses the same technique as the translation from the Horn fragment to first-order logic – principals are made arguments to predicates that occur under their statements. We believe that the above translation to first-order logic justifies the logical correctness of this implementation technique.

Soutei [49] is a dialect of Binder that uses goal-directed search instead of the saturating search. Soutei is a proper subset of the Horn fragment of BL_0, because it does not allow says on clause heads that Binder does, and its provability coincides with that of BL_0. As a result, Theorem 5.1 provides formal evidence that Soutei is indeed an implementation of Binder, and that (on their common fragment) they authorize the same accesses.

6 Implementation and Open Issues

We have implemented the goal-directed search for HH described in Section 3. Our implementation also supports a hybrid modality to represent time explicitly [24], constraints, and system state, and it produces explicit proof-terms. The implementation is less than 500

lines of SML code, and follows a generic structure for logic programming engines described by Pfenning and Elliott [25]. The implementation is quite fast. For example, in the case of policies for controlling classified information, all proofs that we have seen so far can be constructed in less than 300ms on a 2.4GHz Core 2 Duo machine. Some of these proofs are quite large, containing approximately 1100 rule applications and nearly 70 hypotheses. Further, these figures are conservative because we have not yet implemented any optimizations such as residuation and term indexing.

We have also implemented goal-directed search as well as a proof verifier for BL_0 in the logical framework Twelf [47]. We provide online both our implementations and the full case study from which the examples of this paper are derived [28].

Open Issues and Future Work. A very relevant issue that we have not addressed here is termination. For most applications, it is useful to have a guarantee that the prover will terminate. It seems likely that mode and termination checking (for e.g., as implemented in Twelf) can be used to statically rule out policies on which goal-directed search may not terminate. For saturating search, it is hard to prove termination, except on simple fragments such as Datalog. It may be interesting and extremely useful to study this problem further.

Another possibility not considered here is meaningful combination of backward and forward chaining in the same procedure. This is beneficial because predicates whose proofs are reused often can be forward chained and others can be backward chained to improve efficiency. This kind of a mixed search strategy is known to be complete for intuitionistic logic [20], and we expect that the same results will extend to BL_0. In ongoing work, we are also extending the results of this paper to the larger logic BL, which contains both explicit time and constraints. The interaction between the two is non-trivial, especially in the case of saturating search.

7 Conclusion

We have introduced BL_0, an authorization logic with an unusual but useful says modality, and identified fragments of the logic that admit complete goal-directed and saturating proof search strategies. Given that these strategies exist, the next step would be to apply them to meaningful problems. We are currently considering many such applications including an implementation of proof-carrying authorization in a file system, which uses goal-directed search, and policy compilation to low level permissions that uses saturating search.

Acknowledgments. The author wishes to acknowledge Frank Pfenning for continuous support and feedback during the course of this research, and Anton Bachin for pointing out several typographical errors in earlier versions of the paper.

References

[1] SecPAL research release for .NET, 2007. Online at http://research.microsoft.com/en-us/projects/secpal/.

[2] Martín Abadi. Logic in access control. In *Proceedings of the 18th Annual Symposium on Logic in Computer Science (LICS'03)*, pages 228–233, June 2003.

[3] Martín Abadi. Access control in a core calculus of dependency. *Electronic Notes in Theoretical Computer Science*, 172:5–31, apr 2007. *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin.*

[4] Martín Abadi. Variations in access control logic. In *Ninth International Conference on Deontic Logic in Computer Science (DEON 2008)*, pages 96–109, 2008.

[5] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

[6] Natasha Alechina, Michael Mendler, Valeria de Paiva, and Eike Ritter. Categorical and Kripke semantics for constructive S4 modal logic. In *CSL '01: Proceedings of the 15th International Workshop on Computer Science Logic*, pages 292–307, London, UK, 2001. Springer-Verlag.

[7] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[8] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In G. Tsudik, editor, *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 52–62, Singapore, November 1999. ACM Press.

[9] Lujo Bauer. *Access Control for the Web via Proof-Carrying Authorization.* PhD thesis, Princeton University, November 2003.

[10] Lujo Bauer, Scott Garriss, Jonathan M. McCune, Michael K. Reiter, Jason Rouse, and Peter Rutenbar. Device-enabled authorization in the Grey system. In *Information Security: 8th International Conference (ISC '05)*, Lecture Notes in Computer Science, pages 431–445, September 2005.

[11] Lujo Bauer, Scott Garriss, and Michael K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 Symposium on Security and Privacy*, pages 81–95, May 2005.

[12] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. Design and semantics of a decentralized authorization language. In *20th IEEE Computer Security Foundations Symposium*, pages 3–15, 2007.

[13] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management applied to health records. In *Proceedings of IEEE Computer Security Foundations Workshop*, pages 139 – 154, 2004.

[14] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Transactions on Information and Systems Security*, 6(1):71–127, 2003.

[15] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.

[16] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Washington, DC, USA, 1996. IEEE Computer Society.

[17] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. Audit-based compliance control. *International Journal of Information Security*, 6(2):133–151, 2007.

[18] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1989.

[19] Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, 2003.

[20] Kaustuv Chaudhuri, Frank Pfenning, and Greg Price. A logical characterization of forward and backward chaining in the inverse method. *Journal of Automated Reasoning*, 40(2-3):133–177, 2008.

[21] Andrew Cirillo, Radha Jagadeesan, Corin Pitcher, and James Riely. Do As I SaY! Programmatic access control with explicit identities. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF-20)*, pages 16–30, 2007.

[22] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[23] John DeTreville. Binder, a logic-based security language. In M. Abadi and S. Bellovin, editors, *Proceedings of the 2002 Symposium on Security and Privacy (S&P'02)*, pages 105–113, Berkeley, California, May 2002. IEEE Computer Society Press.

[24] Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF-21)*, Pittsburgh, Pennsylvania, June 2008. IEEE Computer Society Press. Extended version available as Technical Report CMU-CS-07-166.

[25] Conal Elliott and Frank Pfenning. A semi-functional implementation of a higher-order logic programming language. In Peter Lee, editor, *Topics in Advanced Language Implementation*, pages 289–325. MIT Press, 1991.

[26] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*. The MIT Press, 1990.

[27] Cédric Fournet, Andrew Gordon, and Sergio Maffeis. A type discipline for authorization in distributed systems. In *CSF '07: Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 31–48. IEEE Computer Society, 2007.

[28] Deepak Garg. BL implementation and case study. Online at http://www.cs.cmu.edu/~dg/bl.

[29] Deepak Garg. Principal-centric reasoning in constructive authorization logic, 2008. Workshop on Intuitionistic Modal Logic and Applications (IMLA'08). Full version available as technical report CMU-CS-09-120.

[30] Deepak Garg and Martín Abadi. A modal deconstruction of access control logics. In *Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2008)*, pages 216–230, Budapest, Hungary, April 2008.

[31] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In J. Guttman, editor, *Proceedings of the 19th Computer Security Foundations Workshop (CSFW '06)*, pages 283–293, Venice, Italy, July 2006. IEEE Computer Society Press.

[32] Yuri Gurevich and Itay Neeman. DKAL: Distributed-knowledge authorization language. In *Proceedings of the 21st IEEE Symposium on Computer Security Foundations (CSF-21)*, 2008.

[33] Yuri Gurevich and Itay Neeman. The logic of infons. Technical report, Microsoft Research, 2009.

[34] Joshua S. Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[35] Limin Jia, Jeffrey A. Vaughan, Karl Mazurak, Jianzhou Zhao, Luke Zarko, Joseph Schorr, and Steve Zdancewic. Aura: A programming language for authorization and audit. In *Proceedings of the International Conference on Functional Programming (ICFP)*, 2008.

[36] Trevor Jim. SD3: A trust management system with certified evaluation. In *Proceedings of IEEE Symposium on Security and Privacy*, 2001.

[37] Chris Lesniewski-Laas, Bryan Ford, Jacob Strauss, Robert Morris, and M. Frans Kaashoek. Alpaca: Extensible authorization for distributed services. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS-2007)*, Alexandria, VA, October 2007.

[38] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73. Springer-Verlag, 2003.

[39] Ninghui Li, John C. Mitchell, and W.H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114 – 130, 2002.

[40] Ninghui Li, John C. Mitchell, and William H. Winsborough. Beyond proof-of-compliance: security analysis in trust management. *Journal of ACM*, 52(3):474–514, 2005.

[41] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.

[42] Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In *Proceedings of the 7th International Symposium on Principles and Practice of Declarative Programming (PPDP'05)*, Lisbon, Portugal, 2005.

[43] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[44] Mehmet A. Orgun. Temporal and modal logic programming: an annotated bibliography. *SIGART Bulletin*, 5(3):52–59, 1994.

[45] Frank Pfenning. Structural cut elimination I. Intuitionistic and classical logic. *Information and Computation*, 157(1/2):84–141, March 2000.

[46] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001.

[47] Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.

[48] Frank Pfenning and Robert J. Simmons. Substructural operational semantics as ordered logic programming. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2009.

[49] Andrew Pimlott and Oleg Kiselyov. Soutei, a logic-based trust-management system. In *Proceedings of the Eighth International Symposium on Functional and Logic Programming (FLOPS 2006)*, pages 130–145, 2006.

[50] R. Sekar, I. V. Ramakrishnan, and Andrei Voronkov. Term indexing. In *Handbook of automated reasoning*, pages 1853–1964. Elsevier Science Publishers B. V., 2001.

[51] Jeffrey A. Vaughan, Limin Jia, Karl Mazurak, and Steve Zdancewic. Evidence-based audit. In *Proceedings of the 21st IEEE Symposium on Computer Security Foundations (CSF-21)*, 2008.

[52] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.

[53] Ted Wobber, Aydan Yumerefendi, Martín Abadi, Andrew Birrell, and Daniel R. Simon. Authorizing applications in singularity. In *EuroSys '07: Proceedings of the 2007 conference on EuroSys*, pages 355–368, New York, NY, USA, 2007. ACM Press.

A Proof of Theorem 3.1

We summarize the basic steps in the proofs of soundness and completeness of goal-directed search on HH.

Theorem A.1 (Soundness)**.** *The following hold.*

1. $\Sigma; \Delta \stackrel{k}{\Rightarrow} g$ *implies* $\Sigma; \Delta \stackrel{k}{\longrightarrow} g$
2. $\Sigma; \Delta; \Xi \stackrel{k}{\Leftarrow} g$ *implies* $\Sigma; \Delta, \Xi \stackrel{k}{\longrightarrow} g$
3. $\Sigma; \Delta \stackrel{k}{\Leftrightarrow} p$ *implies* $\Sigma; \Delta \stackrel{k}{\longrightarrow} p$

4. $\Sigma; d \ll p \mid g_1 \dots g_n$, $d \in \Delta$, *and* $\Sigma; \Delta \xrightarrow{k} g_i$ *imply* $\Sigma; \Delta \xrightarrow{k} p$

Proof. We first prove (4) by induction on the derivation of $\Sigma; d \ll p \mid g_1 \dots g_n$. (1), (2), and (3) follow by simultaneous induction on the given derivations, using (4) in the cases of rules (B1) and (B2). □

Completeness of goal-directed search is harder to establish. We define and prove invertibility of rules in the sequent calculus, and use that to prove completeness.

Definition. A connective is called strongly invertible on the left if the premises of all its left sequent calculus rules (without the principal formulas) can be established by derivations of smaller or equal size whenever their conclusions can be.[2]

Lemma A.2 (Invertibility)**.** *The following hold for the sequent calculus in Figure 2.*

1. *The connectives* $\vee, \exists, \wedge, \mathsf{says}$ *are strongly invertible on the left.*
2. *If* $\Sigma; \Delta \xrightarrow{k} p$, *then there is a clause* d *and formulas* $g_1 \dots g_n$ *such that*
   (a) *Either* $d \in \Delta$ *or for some* k', k' claims $d \in \Delta$ *and* $k' \succeq k$.
   (b) $\Sigma; d \ll p \mid g_1 \dots g_n$.
   (c) *For* $1 \leq i \leq n$ *it is the case that* $\Sigma; \Delta \xrightarrow{k} g_i$ *by derivations of strictly smaller size.*

Proof. (1) follows by a separate induction on given derivations for each connective. (2) follows by an induction on the given derivation of $\Sigma; \Delta \xrightarrow{k} p$. □

Theorem A.3 (Completeness)**.** *The following hold.*

1. *If* $\Sigma; \Delta \xrightarrow{k} g$, *then* $\Sigma; \Delta \overset{k}{\Rightarrow} g$
2. *If* $\Sigma; \Delta, \Xi \xrightarrow{k} g$, *then* $\Sigma; \Delta; \Xi \overset{k}{\Leftarrow} g$
3. *If* $\Sigma; \Delta \xrightarrow{k} p$, *then* $\Sigma; \Delta \overset{k}{\Leftrightarrow} p$

Proof. By a simultaneous lexicographic induction, first on the size of the given sequent calculus derivations and then on the order (2) > (1) > (3). For (2), we subinduct on the number of top level connectives says, $\wedge$, $\vee$, $\top$, and $\exists$ in Ξ. The proof proceeds through a case analysis on the last rule of the given derivation in (1), a case analysis on the structure of formulas in Ξ in (2) using Lemma A.2(1), and directly by Lemma A.2(2) in (3). □

[2] The size of a derivation is defined to be the number of inference rules in the derivation. Note that our definition of invertibility is unusual in that it refers to the size of derivations, not the more conventional depth.

B Proof of Theorem 4.1

In this appendix, we outline a proof of completeness of saturating search. Soundness is easy to prove, we only need to induct on derivations of Figure 4. To prove that saturating search on FHH is complete, we construct an intermediate *weak focusing* system (WF) for FHH. Then we show that any sequent calculus proof (over the FHH fragment) can be simulated in WF, and finally show that any proof in WF can be simulated in FHH. WF is similar to the saturating system in Figure 4, but does not contain L sequents. Instead it allows left rules on chunks to be applied at any time. This technique is adapted from work by Pfenning and Simmons [48].

Weak focusing system. The syntax of formulas for WF is the same as FHH. The hypotheses Δ and Ξ are merged into one set, which we denote Ψ.

$$\begin{array}{llll} \text{Hyps} & \Psi & ::= & \cdot \mid \Psi, k \ \mathsf{claims}\ p \mid \Psi, d \mid \Psi, h \\ \text{Sequents} & R & ::= & \Sigma; \Psi \stackrel{k}{\Longrightarrow} [g] \\ & N & ::= & \Sigma; \Psi \stackrel{\ell}{\Longrightarrow} g \\ & F & ::= & \Sigma; \Psi, [d] \stackrel{\ell}{\Longrightarrow} g \end{array}$$

We use three judgments R (right focus), N (neutral), F (left focus). Instead of using different sequent arrows, we indicate the type of sequent by putting the formula in focus in square brackets $[\cdot]$. The rules are shown in Figure 6. WF admits some obvious structural rules, that we do not state explicitly.

From the Sequent Calculus to WF. First, we prove that any proof in the sequent calculus can be simulated in WF. In order to do that, we need to show that most rules of sequent calculus are admissible in N sequents in WF.

Lemma B.1 (Rule admissibility)**.** *The following hold for WF.*

1. k' $\mathsf{claims}\ p \in \Psi$ *and* $k' \succeq k$ *imply* $\Sigma; \Psi \stackrel{k}{\Longrightarrow} p$
2. $\Sigma; \Psi| \stackrel{k}{\Longrightarrow} g$ *implies* $\Sigma; \Psi \stackrel{k'}{\Longrightarrow} k \ \mathsf{says}\ g$
3. $\Sigma; \Psi_1 \stackrel{k}{\Longrightarrow} g_1$ *and* $\Sigma; \Psi_2 \stackrel{k}{\Longrightarrow} g_2$ *imply* $\Sigma; \Psi_1, \Psi_2 \stackrel{k}{\Longrightarrow} g_1 \wedge g_2$
4. $\Sigma; \Psi \stackrel{k}{\Longrightarrow} g_1$ *implies* $\Sigma; \Psi \stackrel{k}{\Longrightarrow} g_1 \vee g_2$
5. $\Sigma; \Psi \stackrel{k}{\Longrightarrow} g_2$ *implies* $\Sigma; \Psi \stackrel{k}{\Longrightarrow} g_1 \vee g_2$
6. $\Sigma, x{:}\sigma; \Psi \stackrel{k}{\Longrightarrow} g$ *implies* $\Sigma; \Psi \stackrel{k}{\Longrightarrow} \forall x{:}\sigma.g$
7. $\Sigma; \Psi \stackrel{k}{\Longrightarrow} g[t/x]$ *and* $\Sigma \vdash t{:}\sigma$ *imply* $\Sigma; \Psi \stackrel{k}{\Longrightarrow} \exists x{:}\sigma.g$
8. $\Sigma; \Psi, d_1, d_2, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} g$ *implies* $\Sigma; \Psi, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} g$

R sequents

$$\frac{k' \text{ claims } p \in \Psi \qquad k' \succeq k}{\Sigma; \Psi \overset{k}{\Longrightarrow} [p]}\text{init} \qquad \frac{\Sigma; \Psi| \overset{k}{\Longrightarrow} [g]}{\Sigma; \Psi \overset{k'}{\Longrightarrow} [k \text{ says } g]}\text{saysR}$$

$$\frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g] \qquad \Sigma; \Psi \overset{k}{\Longrightarrow} [g']}{\Sigma; \Psi \overset{k}{\Longrightarrow} [g \wedge g']}\wedge\text{R} \qquad \frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g]}{\Sigma; \Psi \overset{k}{\Longrightarrow} [g \vee g']}\vee \text{R}_1 \qquad \frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g']}{\Sigma; \Psi \overset{k}{\Longrightarrow} [g \vee g']}\vee \text{R}_2$$

$$\frac{}{\Sigma; \Psi \overset{k}{\Longrightarrow} [\top]}\top\text{R} \qquad \frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g[t/x]] \qquad \Sigma \vdash t{:}\sigma}{\Sigma; \Psi \overset{k}{\Longrightarrow} [\exists x{:}\sigma.g]}\exists\text{R}$$

N sequents

$$\frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g]}{\Sigma; \Psi \overset{k}{\Longrightarrow} g}\text{F1} \qquad \frac{d \in \Psi \qquad \Sigma; \Psi, [d] \overset{k}{\Longrightarrow} g}{\Sigma; \Psi \overset{k}{\Longrightarrow} g}\text{F2} \qquad \frac{\Sigma; \Psi, k' \text{ claims } p \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, k' \text{ says } p \overset{k}{\Longrightarrow} g}\text{saysL}$$

$$\frac{\Sigma; \Psi, h, h' \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, h \wedge h' \overset{k}{\Longrightarrow} g}\wedge\text{L} \qquad \frac{\Sigma; \Psi \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, \top \overset{k}{\Longrightarrow} g}\top\text{L} \qquad \frac{}{\Sigma; \Psi, \bot \overset{k}{\Longrightarrow} g}\bot\text{L}$$

$$\frac{\Sigma; \Psi, h \overset{k}{\Longrightarrow} g \qquad \Sigma; \Psi, h' \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, h \vee h' \overset{k}{\Longrightarrow} g}\vee\text{L} \qquad \frac{\Sigma, x{:}\sigma; \Psi, h \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, \exists x{:}\sigma.h \overset{k}{\Longrightarrow} g}\exists\text{L}$$

F sequents

$$\frac{\Sigma; \Psi, h \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, [h] \overset{k}{\Longrightarrow} g}\text{blur} \qquad \frac{\Sigma; \Psi, [d_1] \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, [d_1 \wedge d_2] \overset{k}{\Longrightarrow} g}\wedge\text{L}_1 \qquad \frac{\Sigma; \Psi, [d_2] \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, [d_1 \wedge d_2] \overset{k}{\Longrightarrow} g}\wedge\text{L}_2$$

$$\frac{\Sigma; \Psi \overset{k}{\Longrightarrow} [g_1] \qquad \Sigma; \Psi, [d_2] \overset{k}{\Longrightarrow} g}{\Sigma; \Psi, [g_1 \supset d_2] \overset{k}{\Longrightarrow} g}\supset\text{L} \qquad \frac{\Sigma; \Psi, [d[t/x]] \overset{k}{\Longrightarrow} g \qquad \Sigma \vdash t{:}\sigma}{\Sigma; \Psi, [\forall x{:}\sigma.d] \overset{k}{\Longrightarrow} g}\forall\text{L}$$

Figure 6: WF: Weak focusing system for FHH

9. $\Sigma; \Psi_1, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g_1$ *and* $\Sigma; \Psi_2, d_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g$ *imply* $\Sigma; \Psi_1, \Psi_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g$.

10. $\Sigma; \Psi, \forall x{:}\sigma.d, d[t/x] \stackrel{k}{\Longrightarrow} g$ *and* $\Sigma \vdash t{:}\sigma$ *imply* $\Sigma; \Psi, \forall x{:}\sigma.d \stackrel{k}{\Longrightarrow} g$.

Proof.

1. We have the following direct derivation

$$\dfrac{\dfrac{k' \ \mathsf{claims}\ p \in \Delta \qquad k' \succeq k}{\Sigma; \Psi \stackrel{k}{\Longrightarrow} [p]}\ \text{INIT}}{\Sigma; \Psi \stackrel{k}{\Longrightarrow} p}\ \text{F1}$$

2. The only rule that may derive $\Sigma; \Psi| \stackrel{k}{\Longrightarrow} g$ is F1. This is because $\Psi|$ only has assumptions of the form k' claims p, so no left rule applies. Hence it follows that we must have $\Sigma; \Psi| \stackrel{k}{\Longrightarrow} [g]$ (premise of F1). By rule (saysR), $\Sigma; \Psi \stackrel{k'}{\Longrightarrow} [k\ \mathsf{says}\ g]$, and by F1, $\Sigma; \Psi \stackrel{k'}{\Longrightarrow} k\ \mathsf{says}\ g$.

3. We generalize the statement of (3), adding the following two new hypothesis:

   (a) $\Sigma; \Psi_1 \stackrel{k}{\Longrightarrow} g_1$ and $\Sigma; \Psi_2, [d] \stackrel{k}{\Longrightarrow} g_2$ imply $\Sigma; \Psi_1, \Psi_2, [d] \stackrel{k}{\Longrightarrow} g_1 \wedge g_2$

   (b) $\Sigma; \Psi_1, [d] \stackrel{k}{\Longrightarrow} g_1$ and $\Sigma; \Psi_2 \stackrel{k}{\Longrightarrow} g_2$ imply $\Sigma; \Psi_1, \Psi_2, [d] \stackrel{k}{\Longrightarrow} g_1 \wedge g_2$

   All three statements follow by a simultaneous induction on the two given derivations.

4. We generalize the statement of (4), adding the following new hypothesis:

   (a) $\Sigma; \Psi, [d] \stackrel{k}{\Longrightarrow} g_1$ implies $\Sigma; \Psi, [d] \stackrel{k}{\Longrightarrow} g_1 \vee g_2$

   Both statements follow by a simultaneous induction on given derivations.

5. Same technique as (4).

6. Same technique as (4).

7. Same technique as (4).

8. We generalize the statement of (8), adding the following new hypothesis:

   (a) $\Sigma; \Psi, d_1, d_2, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} [g]$ implies $\Sigma; \Psi, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} [g]$

   (b) $\Sigma; \Psi, d_1, d_2, d_1 \wedge d_2, [d] \stackrel{k}{\Longrightarrow} g$ implies $\Sigma; \Psi, d_1 \wedge d_2, [d] \stackrel{k}{\Longrightarrow} g$

(a) follows by induction on the given derivation. Next, we prove (8) and (b) by simultaneous induction on given derivations, using (a) for the case of rule (F1). The only other interesting case is when the given derivation is the following:

$$\frac{\Sigma; \Delta, d_1, d_2, d_1 \wedge d_2, [d_1] \stackrel{k}{\Longrightarrow} g}{\Sigma; \Delta, d_1, d_2, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} g}\text{F2}$$

By i.h. on premise, $\Sigma; \Delta, d_1 \wedge d_2, [d_1] \stackrel{k}{\Longrightarrow} g$. By rule ($\wedge$ L$_1$), $\Sigma; \Delta, d_1 \wedge d_2, [d_1 \wedge d_2] \stackrel{k}{\Longrightarrow} g$. By rule (F2), $\Sigma; \Delta, d_1 \wedge d_2 \stackrel{k}{\Longrightarrow} g$.

9. We generalize the statement of (9), adding the following hypothesis

   (a) $\Sigma; \Psi, d_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} [g]$ implies $\Sigma; \Psi, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} [g]$.

   (b) $\Sigma; \Psi_1, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} [g_1]$ and $\Sigma; \Psi_2, d_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g$ imply $\Sigma; \Psi_1, \Psi_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g$.

   (c) $\Sigma; \Psi_1, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} [g_1]$ and $\Sigma; \Psi_2, d_2, g_1 \supset d_2, [d] \stackrel{k}{\Longrightarrow} g$ imply $\Sigma; \Psi_1, \Psi_2, g_1 \supset d_2, [d] \stackrel{k}{\Longrightarrow} g$.

   (d) $\Sigma; \Psi_1, g_1 \supset d_2, [d] \stackrel{k}{\Longrightarrow} g_1$ and $\Sigma; \Psi_2, d_2, g_1 \supset d_2 \stackrel{k}{\Longrightarrow} g$ imply $\Sigma; \Psi_1, \Psi_2, g_1 \supset d_2, [d] \stackrel{k}{\Longrightarrow} g$.

   (a) follows by induction on the given derivation. (b) and (c) follow by simultaneous induction on the second given derivations, using (a) when the derivation ends in rule (F1). (9) and (d) follow by simultaneous induction on the first given derivation, using (b) and (c) when the derivation ends in rule (F1).

10. We generalize the statement of (10), adding the following two new hypothesis:

    (a) $\Sigma; \Psi, \forall x{:}\sigma.d, d[t/x] \stackrel{k}{\Longrightarrow} [g]$ and $\Sigma \vdash t{:}\sigma$ imply $\Sigma; \Psi, \forall x{:}\sigma.d \stackrel{k}{\Longrightarrow} [g]$.

    (b) $\Sigma; \Psi, \forall x{:}\sigma.d, d[t/x], [d'] \stackrel{k}{\Longrightarrow} g$ and $\Sigma \vdash t{:}\sigma$ imply $\Sigma; \Psi, \forall x{:}\sigma.d, [d'] \stackrel{k}{\Longrightarrow} g$.

    (a) follows by induction on the given derivation. (10) and (b) then follow by a simultaneous induction on given derivations, using (a) for the case of rule (F1).

□

We can now prove that on the FHH fragment of BL$_0$, the sequent calculus can be simulated in WF. We define a class of *regular sequents* as follows.

Definition. A sequent $\Sigma; \Gamma \stackrel{k}{\longrightarrow} s$ is called regular if:

1. s is an FHH goal g.
2. Γ contains assumptions of the following forms only

(a) FHH clauses d

(b) FHH chunks h

(c) k' claims p

(d) p, where for some $k' \succeq k$, k' claims $p \in \Gamma$.

Further if Γ is regular, we define $\Gamma\uparrow$ to be its largest subset that does not contain any atoms. Note that $\Gamma\uparrow$ is always a valid hypotheses in WF (i.e., it is of the form Ψ).

Lemma B.2 (Weak completeness)**.** *If* $\Sigma; \Gamma \xrightarrow{k} s$ *is regular and provable in* BL_0*'s sequent calculus, then* $\Sigma; \Gamma\uparrow \overset{k}{\Longrightarrow} s$ *in WF.*

Proof. We induct on the depth of the derivation of $\Sigma; \Gamma \xrightarrow{k} s$. We case analyze the last rule of the derivation. For the rules ($\vee$ L), ($\exists$L), and (says L) and ($\wedge$ L) applied to a chunk, we first appeal to Lemma A.2.3, apply the i.h. to the smaller derivation, and then apply the corresponding rule in WF. Case ($\bot$L) follows by the corresponding rule in WF. Case ($\top$R) follows by rules ($\top$R) and (F1) in WF. For ($\wedge$ L) applied to a clause, we use Lemma B.1.8. The remaining cases except (init) and (claims) also follow from Lemma B.1. The cases (init) and (claims) are shown below.

Case. $\dfrac{}{\Sigma; \Gamma, p \xrightarrow{k} p}\text{init}$

By the assumption of regularity, for some $k' \succeq k$, k' claims $p \in \Gamma$. Clearly, k' claims $p \in \Gamma\uparrow$. Hence by Lemma B.1.1, $\Sigma; \Gamma\uparrow \overset{k}{\Longrightarrow} p$.

Case. $\dfrac{k \succeq k_0 \qquad \Sigma; \Gamma, k\ \mathsf{claims}\ s, s \xrightarrow{k_0} r}{\Sigma; \Gamma, k\ \mathsf{claims}\ s \xrightarrow{k_0} r}\text{claims}$

By assumption of regularity, s must be an atom. It follows that $(\Gamma, k\ \mathsf{claims}\ s, s)\uparrow = (\Gamma, k\ \mathsf{claims}\ s)\uparrow$. Therefore, by i.h. on the premise, $\Sigma; (\Gamma, k\ \mathsf{claims}\ s)\uparrow \overset{k_0}{\Longrightarrow} r$. □

From WF to FHH. Next we show that every proof in WF can be simulated in FHH. Completeness of saturating search follows immediately from this and the preceding lemma.

Lemma B.3 (WF to FHH)**.** *The following hold for WF and FHH.*

1. $\Sigma; \Delta, \Xi \overset{k}{\Longrightarrow} g$ *implies* $\Sigma; \Delta; \Xi \overset{k}{\Leftarrow} g$
2. $\Sigma; \Delta \overset{k}{\Longrightarrow} g$ *implies* $\Sigma; \Delta \overset{k}{\Uparrow} g$
3. $\Sigma; \Delta, [d] \overset{k}{\Longrightarrow} g$ *implies that there exist* $g_1 \ldots g_n$ *and* h *such that* $\Sigma; d \ll h \mid g_1 \ldots g_n$, $\Sigma; \Delta \overset{k}{\gg} g_i$ *and* $\Sigma; \Delta; h \overset{k}{\Leftarrow} g$
4. $\Sigma; \Delta \overset{k}{\Longrightarrow} [g]$ *implies* $\Sigma; \Delta \overset{k}{\gg} g$

Proof. All statements follow by a simultaneous induction on given derivations. □

Theorem B.4 (Completeness)**.** *If* $\Sigma; \Delta \xrightarrow{k} g$ *in* BL_0*'s sequent calculus, then* $\Sigma; \Delta \overset{k}{\Leftrightarrow} g$ *in FHH.*

Proof. Assume that $\Sigma; \Delta \xrightarrow{k} g$. Since $\Sigma; \Delta \xrightarrow{k} g$ is always a regular sequent, and $\Delta \uparrow = \Delta$, by Lemma B.2 we get $\Sigma; \Delta \overset{k}{\Longrightarrow} g$ in WF. Hence by Lemma B.3.2, $\Sigma; \Delta \overset{k}{\Leftrightarrow} g$ in FHH. □