Labeled Goal-directed Search in Access Control Logic

Valerio Genovese[1,2], Deepak Garg[3], and Daniele Rispoli[2]

[1] University of Luxembourg, Luxembourg
[2] University of Torino, Torino, Italy
[3] MPI-SWS, Kaiserslautern and Saarbrücken, Germany

Abstract. We describe a sound, complete, and terminating procedure for goal-directed proof search in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$, an expressive fragment of a recently presented access control logic, $\mathrm{BL}_{\mathrm{sf}}$. $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is more expressive than many other Datalog-based access control logics that also have very efficient decision procedures, and our search procedure finds proofs of authorization quickly in practice. We also extend our search procedure to find missing credentials when a requested authorization does not hold and discuss an implementation of our techniques in an extension of the Unix file synchronization program `rsync`.

Keywords: Access control, formal logic, goal-directed search, labeled calculus

1 Introduction

Many access control systems rely on representation of authorization policies as logical theories [11,4,5,3,16,21]. Such representation not only formalizes high-level policy intent through the logic's semantics, but also allows for direct enforcement of policies through architectures like proof-carrying authorization [11,4,5], as well as formal proofs of policy meta properties like non-interference [10,7]. In fact, a number of special modal logics, called access control logics or authorization logics, have been proposed specifically for representing, enforcing and reasoning about access policies [2,10,11,12,20,8]. Policy representation in logic and enforcement are related as follows: A requested access, represented as the logical formula φ, is authorized by a policy represented as the logical theory Γ iff Γ entails φ. Consequently, to enforce policies represented in logic through a computer system, it is important to have a method to efficiently check entailment in the logic or, equivalently, to have an efficient theorem prover for the logic. It is also useful, but not necessary, to know that the prover terminates in all cases without losing completeness (implying that the logic is decidable), so the theorem prover can be invoked in a reference monitor without having to worry about non-termination. There is a tradeoff between designing an access control logic with many useful logical connectives and one with an efficiently implementable decision procedure.

In prior work [14] (called EW or earlier work in the sequel), we have shown that an expressive access control logic, $\mathrm{BL}_{\mathrm{sf}}$, is decidable and that its decision procedure can be used as a foundation to solve three other practical problems in access control: justifying denied access (countermodel construction), finding all consequences of a policy (saturation), and determining what additional credentials will allow a denied access (policy abduction). Although an interesting theoretical result, experimental evaluation of the EW decision procedure (explained in Section 2) indicates that it has at least exponential complexity even on very simple access policies. While this is not surprising given the generality of the decision procedure, the problem we investigate now is that of reducing this complexity in practice, albeit at the cost of sacrificing some of the decision procedure's applications.

Precisely, we argue, through theoretical and experimental results, that it is pragmatic to consider a restriction of $\mathrm{BL}_{\mathrm{sf}}$ to what is called a *Hereditary-Harrop (HH) fragment* [17] and to use *goal-directed* theorem proving on it. The HH fragment of logic, first considered in λ-Prolog, is a generalization of the Horn fragment of first-order logic on which Prolog is based (and, in our case, further generalized to include access control-specific connectives of $\mathrm{BL}_{\mathrm{sf}}$). Goal-directed proof search, also called SLD resolution or top-down search in logic programming, is an efficient technique for theorem proving that prunes search very rapidly by selecting only those rules from the theory that can directly prove the goal at hand. Completeness of this pruning relies heavily on restriction to the Horn or HH fragment.

We first define the HH fragment of $\mathrm{BL}_{\mathrm{sf}}$, called $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ (the G in the superscript stands for goal-directed). We argue by examples that although only a fragment of $\mathrm{BL}_{\mathrm{sf}}$, $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ is very expressive and can represent policies that cannot be represented in other restrictions considered in literature to attain efficient proof search, e.g., the Datalog fragment. Second, we describe a goal-directed proof search procedure for $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ and prove that it is sound and complete. Following EW, our procedure is based in labeled sequent calculi [18,22], which directly use the *semantic* definitions of the logic's connectives in proof rules. Although our completeness proof follows a standard template, it is a non-trivial generalization of existing proofs to account for the labeled style and also to accommodate two access control-specific constructs of $\mathrm{BL}_{\mathrm{sf}}$ — A says φ and A sf B (the latter means that principal A speaks for principal B). We then show by a careful counting of steps in our completeness proof that the termination condition of EW translates into a uniform bound on the depth of search required in $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, thus implying that our goal-directed search procedure is a decision procedure. Although the worst-case bound of this procedure is not good in theory, we explain why goal-directed search works very efficiently in practice and confirm this explanation experimentally.

Next, we show how our goal-directed search procedure can be adapted to also find missing credentials when the policy does not allow an access. This

adaptation, called an abduction procedure, mirrors a similar procedure in EW.[1] Finally, we discuss a practical application of our work: We design and implement an extension of the `rsync` software for remote file synchronization, allowing rich file access policies written in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ and relying on our abduction procedure for automatically obtaining credentials from online servers when access is denied.

Besides describing and justifying the use of goal-directed proof search for $\mathrm{BL}_{\mathrm{sf}}$, we make two minor technical contributions to goal-directed search in general: (1) To the best of our knowledge, we present the first goal-directed labeled calculi for the HH fragment of a multimodal intuitionistic logic and (2) Our counting argument to establish termination bounds on goal-directed search through the completeness proof is novel and somewhat surprising in goal-directed literature.

The paper is organized as follows. In Section 2, we recall the logic $\mathrm{BL}_{\mathrm{sf}}$ from EW, the method of labeled sequent calculi and EW's decision procedure, and explain by experimental evaluation the exponential behavior of the EW search procedure even on simple access policies. In Section 3, we present $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ together with its goal-directed search procedure. In Section 4, we present our theoretical results, showing that goal-directed search is sound and complete and deriving a depth bound on it. Section 5 presents the extension of goal-directed search to perform abduction and its implementation. Section 6 presents the extension of `rsync` with $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ for representing policies and abduction for finding authorization credentials on-the-fly. Section 7 discusses related work and Section 8 concludes the paper. Due to space constraints, we relegate proofs and other technical details to a technical report [13].

2 Recap of $\mathrm{BL}_{\mathrm{sf}}$: Semantics and Proof-theory

In this section we briefly describe the logic $\mathrm{BL}_{\mathrm{sf}}$ as presented in our earlier work (EW) [14]. $\mathrm{BL}_{\mathrm{sf}}$ is a propositional intuitionistic logic enriched with two well-known connectives in access control logics: A says φ (principal A supports formula φ) and A sf B (principal A speaks for principal B). The former is used to represent assertions of principals and the latter is used to represent trust relationships between principals (A sf B represents that B trusts A). The syntax of $\mathrm{BL}_{\mathrm{sf}}$ formulas is shown below. p denotes an atomic formula, drawn from a countable set of symbols, and A, B denote principals drawn from a different, finite set $\mathcal{I}$. The connectives $\top$ (true), $\bot$ (false), $\wedge$ (and), $\vee$ (or) and $\rightarrow$ (implication) have usual meanings from intuitionistic logic.

$$\text{Formulas } \varphi, \psi ::= p \mid \top \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid A \ \mathsf{says} \ \varphi \mid A \ \mathsf{sf} \ B$$

Syntactically, the logic $\mathrm{BL}_{\mathrm{sf}}$ is characterized by the following axioms:

[1] The other two applications of $\mathrm{BL}_{\mathrm{sf}}$'s decision procedure in EW, namely counter-model construction and saturation, are fundamentally incompatible with pruning employed by goal-directed proof search. Consequently, these two applications are not considered in this paper.

(All intuitionistic propositional tautologies)

$$\frac{\vdash \varphi}{\vdash A \text{ says } \varphi} \quad \text{(nec)}$$

$$\vdash (A \text{ says } (\varphi \to \psi)) \to ((A \text{ says } \varphi) \to (A \text{ says } \psi)) \quad \text{(K)}$$

$$\vdash (A \text{ says } \varphi) \to (B \text{ says } A \text{ says } \varphi) \quad \text{(I)}$$

$$\vdash (A \text{ sf } B) \to ((A \text{ says } \varphi) \to (B \text{ says } \varphi)) \quad \text{(speaksfor)}$$

$$\vdash A \text{ sf } A$$

$$\vdash (A \text{ sf } B) \to ((B \text{ sf } C) \to (A \text{ sf } C))$$

Rule (nec) and axiom (K) are standard in modal logic. Axiom (I) is needed to accurately model delegation in the logic [1], whereas (speaksfor) characterizes the formula A sf B: If A sf B, then any statement φ that A makes is echoed by B, so the formula A sf B means that A has authority to speak on behalf of B [2].

Kripke Semantics The semantics of $\mathrm{BL}_{\mathrm{sf}}$ are presented in the standard, Kripke style for modal logics. A model contains several points called worlds, which represent possible states of knowledge. Modalities are interpreted using binary accessibility relations on worlds, with one relation S_A for every modality (A says $\cdot$). Intuitionistic implication is modeled using a binary preorder, $\leq$. EW treats the formula A sf B as an atom in the Kripke semantics and validates axioms related to it, e.g., (speaksfor), through conditions on Kripke frames.

Definition 1 (Kripke model) *A Kripke model or, simply, model, $\mathcal{M}$ is a tuple $(W, \leq, \{S_A\}_{A \in \mathcal{I}}, h, sf)$ where: W is a set. Its elements are called worlds. $\leq$ is a preorder on W. For each principal A, S_A is a binary relation on W, called the accessibility relation of principal A. h, called the truth assignment or assignment, is a map from the set of atoms to $\mathcal{P}(W)$. For any atom p, $h(p)$ is the set of worlds where p holds. sf is a map from pairs of principals to $\mathcal{P}(W)$. For any two principals A and B, $sf(A, B)$ is the set of worlds where A sf B holds.*

Let $S_{\mathcal{I}} = \bigcup_{A \in \mathcal{I}} S_A$. For $\mathrm{BL}_{\mathrm{sf}}$*, the class of models is further restricted to those that satisfy the following* frame conditions.

- $\forall x.(x \leq x)$ *(refl)*
- $\forall x, y, z.(((x \leq y) \wedge (y \leq z)) \to (x \leq z))$ *(trans)*
- $\forall x, y, z.(((x \leq y) \wedge (y S_A z)) \to (x S_A z))$ *(mon-S)*
- $\forall x, y, z.(((x S_B y) \wedge (y S_A z)) \to (x S_A z))$ *(I)*
- *If $w \in sf(A, B)$, then for all w', $w S_B w'$ implies $w S_A w'$.* *(basic-sf)*
- *For all A and w, $w \in sf(A, A)$.* *(refl-sf)*
- *If $w \in sf(A, B) \cap sf(B, C)$, then $w \in sf(A, C)$.* *(trans-sf)*
- *If $x \in h(p)$ and $x \leq y$, then $y \in h(p)$.* *(mon)*
- *If $x (\leq \cup S_{\mathcal{I}})^* y$ and $x \in sf(A, B)$, then $y \in sf(A, B)$.* *(mon-sf)*

Properties (refl) and (trans) make $\leq$ a preorder. Property (mon-S) validates axiom (K). Property (I) corresponds to axiom (I). Property (basic-sf) corresponds to axiom (speaksfor). Properties (refl-sf) and (trans-sf) make A sf B reflexive and transitive, respectively. Property (mon) is standard in Kripke models of intuitionistic logics and forces monotonicity of satisfaction. Property (mon-sf) implies that if A sf B holds in a world, then it also holds in all future worlds.

Definition 2 (Satisfaction) *Given a model* $\mathcal{M} = (W, \leq, \{S_A\}_{A \in \mathcal{I}}, h, sf)$ *and a world* $w \in W$*, the satisfaction relation* $\mathcal{M} \models w : \alpha$*, read "the world* w *satisfies formula* α *in model* $\mathcal{M}$*", is defined by induction on* α *as follows (standard clauses for* $\top, \wedge$ *and* $\vee$ *are omitted):*

- $\mathcal{M} \models w : p$ *iff* $w \in h(p)$
- $\mathcal{M} \models w : \alpha \to \beta$ *iff for every* w' *such that* $w \leq w'$ *and* $\mathcal{M} \models w' : \alpha$*, we have* $\mathcal{M} \models w' : \beta$.
- $\mathcal{M} \models w : A \ \mathsf{says} \ \alpha$ *iff for every* w' *such that* wS_Aw'*, we have* $\mathcal{M} \models w' : \alpha$.
- $\mathcal{M} \models w : A \ \mathsf{sf} \ B$ *iff* $w \in sf(A, B)$.

$\mathcal{M} \not\models w : \alpha$ *if it is not the case that* $\mathcal{M} \models w : \alpha$*. A formula* α *is true in a model* $\mathcal{M}$*, written* $\mathcal{M} \models \alpha$*, if for every world* $w \in \mathcal{M}$*,* $\mathcal{M} \models w : \alpha$*. A formula* α *is* valid *in* $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$*, written* $\models \alpha$*, if* $\mathcal{M} \models \alpha$ *for every model* $\mathcal{M}$.

Labeled sequent calculus The central technical idea in EW, on which the entire technical development of EW including $\mathrm{BL}_{\mathrm{sf}}$'s decision procedure rests, is a labeled sequent calculus for $\mathrm{BL}_{\mathrm{sf}}$. Our goal-directed search procedure is a restriction of EW's sequent calculus, so we discuss the sequent calculus is some detail here. Following standard presentations of labeled sequent calculi (also called prefixed calculi), EW's calculus, SeqC, manipulates two types of labeled formulas: *world formulas*, written $x : \varphi$, where x is a symbolic world and φ is a formula of the logic (intuitively meaning that φ holds in world x), and *relation formulas*, representing semantic relations of the form $x \leq y$ and xS_Ay between symbolic worlds.

A sequent of SeqC has the form $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ where Σ is a list of world symbols, $\mathbb{M}$ is a multiset of relation formulas and Γ and Δ are multisets of world formulas. Semantically, the sequent $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ means that "every model which satisfies all labeled formulas of $\Gamma \cup \mathbb{M}$ satisfies at least one labeled formula in Δ"; this is made precise in the following definition.

Definition 3 (Sequent satisfaction and validity) *A model* $\mathcal{M}$ *and a mapping* ρ *from elements of* Σ *to worlds of* $\mathcal{M}$ *satisfy a (possibly non-provable) sequent* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$*, written* $\mathcal{M}, \rho \models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$*, if one of the following holds:*

- *There is an* $xRy \in \mathbb{M}$ *with* $R \in \{\leq\} \cup \{S_A \mid A \in \mathcal{I}\}$ *such that* $\rho(x) \ R \ \rho(y) \notin \mathcal{M}$.
- *There is an* $x : \alpha \in \Gamma$ *such that* $\mathcal{M} \not\models \rho(x) : \alpha$.
- *There is an* $x : \alpha \in \Delta$ *such that* $\mathcal{M} \models \rho(x) : \alpha$.

A model $\mathcal{M}$ *satisfies a sequent* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$*, written* $\mathcal{M} \models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$*, if for every mapping* ρ*, we have* $\mathcal{M}, \rho \models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$*. Finally, a sequent* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ *is valid, written* $\models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$ *if for every model* $\mathcal{M}$*, we have* $\mathcal{M} \models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$.

Selected rules of SeqC are reproduced from EW in Figure 1. The first key point to observe about the calculus is that the rules of each connective (e.g., rule $\to$R for implication) mimic directly the *semantic* definition of satisfaction for the

Axiom Rules

$$\frac{}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Rightarrow y : p, \Delta}\text{init} \qquad \frac{}{\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf}\ B \Rightarrow x : A \ \mathsf{sf}\ B, \Delta}\text{sf}$$

Logical Rules

$$\frac{\Sigma, y; \mathbb{M}, x \leq y; \Gamma, y : \alpha \Rightarrow y : \beta, x : \alpha \to \beta, \Delta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \alpha \to \beta, \Delta} \to\text{R}$$

$$\frac{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : \alpha \to \beta \Rightarrow y : \alpha, \Delta \qquad \Sigma; \mathbb{M}, x \leq y; \Gamma, x : \alpha \to \beta, y : \beta \Rightarrow \Delta}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : \alpha \to \beta \Rightarrow \Delta} \to\text{L}$$

$$\frac{\Sigma, y; \mathbb{M}, xS_A y; \Gamma \Rightarrow y : \alpha, x : A \ \mathsf{says}\ \alpha, \Delta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : A \ \mathsf{says}\ \alpha, \Delta}\mathsf{saysR} \qquad \frac{\Sigma; \mathbb{M}, xS_A y; \Gamma, x : A \ \mathsf{says}\ \alpha, y : \alpha \Rightarrow \Delta}{\Sigma; \mathbb{M}, xS_A y; \Gamma, x : A \ \mathsf{says}\ \alpha \Rightarrow \Delta}\mathsf{saysL}$$

Frame Rules

$$\frac{\Sigma; \mathbb{M}, x \leq y, y \leq z, x \leq z; \Gamma \Rightarrow \Delta}{\Sigma; \mathbb{M}, x \leq y, y \leq z; \Gamma \Rightarrow \Delta}\text{trans} \qquad \frac{\Sigma; \mathbb{M}, x \leq y, yS_A z, xS_A z; \Gamma \Rightarrow \Delta}{\Sigma; \mathbb{M}, x \leq y, yS_A z; \Gamma \Rightarrow \Delta}\text{mon-S}$$

$$\frac{\Sigma; \mathbb{M}, xS_B y, yS_A z, xS_A z; \Gamma \Rightarrow \Delta}{\Sigma; \mathbb{M}, xS_B y, yS_A z; \Gamma \Rightarrow \Delta}\text{I} \qquad \frac{\Sigma; \mathbb{M}, xS_B y, xS_A y; \Gamma, x : A \ \mathsf{sf}\ B \Rightarrow \Delta}{\Sigma; \mathbb{M}, xS_B y; \Gamma, x : A \ \mathsf{sf}\ B \Rightarrow \Delta}\text{basic-sf}$$

Fig. 1. SeqC: EW's labeled sequent calculus for $\mathrm{BL}_{\mathsf{sf}}$, selected rules

connective (Definition 2). Second, the frame rules enforce all conditions (refl)–(mon-sf) on models listed in Definition 1, except the condition (mon) which is implicit in the inference rule (init). We write $\vdash (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$ to mean that $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ has a proof. SeqC is sound and complete with respect to the Kripke semantics.

Theorem 4 (Soundness and Completeness [14]). $\vdash (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$ *has a proof if and only if* $\models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$

Decision procedure Backwards proof search in SeqC may not terminate due to potentially infinite creation of new worlds through the rules ($\to$R) and (saysR). Hence, SeqC is not a decision procedure in itself. The key insight in EW is that despite this fact, suitable (and complex) termination conditions can be imposed on backwards search to make it terminate without losing completeness, thus yielding a decision procedure for $\mathrm{BL}_{\mathrm{sf}}$. (Further, EW describes how countermodels can be extracted when search fails.) The specific termination conditions of the decision procedure are not important for this paper. However, the following two facts about the decision procedure *are* relevant.

First, by an analysis of EW's termination proof we can show that for any sequent that we wish to prove, we can compute a number n such that backwards search in SeqC can be pruned at depth n without losing completeness. This fact was not observed in EW, but it is not difficult to prove by a careful analysis of the termination proof in EW. We use this observation to derive a similar

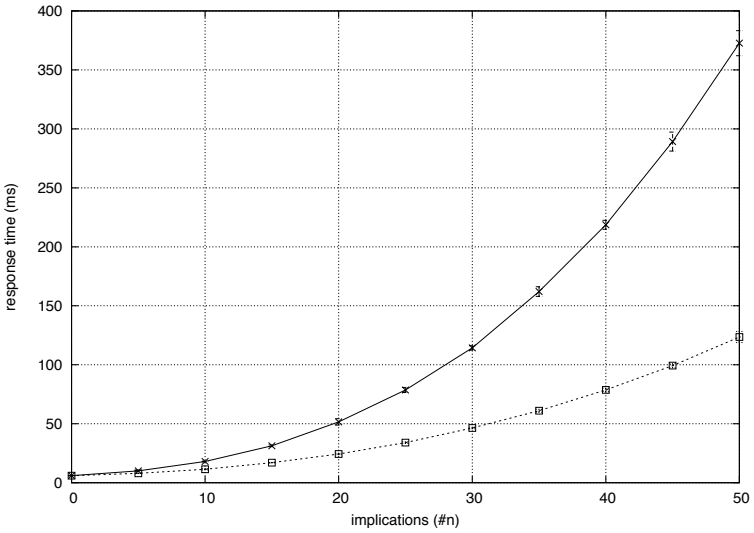

Fig. 2. Scalability of decision procedures based on SeqC and SeqG. The solid line is SeqC and the dotted line is SeqG.

completeness-preserving bound ($3n + 1$ to be exact) for our goal-directed proof calculus, thus implying that our calculus can also be converted to a decision procedure.

Second, the number n is large — it is at least doubly exponential in the size of the sequent. Hence, the worst-case complexity of both our EW's decision procedure for $\mathrm{BL}_{\mathrm{sf}}$ and ours for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is also quite bad. The difference is that on *practical policies*, EW's decision procedure attains at least exponential complexity, whereas our decision procedure remains polynomial (quadratic in many cases). This is a well-known fact from literature on goal-directed search and is, e.g., the reason why Prolog works efficiently in practice; here, we merely exploit this known fact for access control.

To illustrate the second point, we compare the time taken by an implementation of our decision procedure to that taken by EW's decision procedure on a simple, common policy: $\Gamma_n = \{p_1 \to q \wedge p_2 \to p_1 \wedge \ldots \wedge p_n \to p_{n-1}\}$. Here, $p_1, \ldots, p_n$ and q are atoms. The objective is to test the unprovable fact $\Gamma_n \Rightarrow q$. This example is representative of actual access policies, e.g., q may be a proposition representing a permission, p_1 may be condition needed for the permission, which may in turn be contingent upon p_2 and so on. The time taken by EW's procedure and our goal-directed procedures for this example are shown in Figure 2. The upper solid line, corresponding to EW, is exponential in n; the lower dotted line, corresponding to our goal-directed search, is quadratic in n. The reason for the difference is straightforward: EW's procedure works by blindly decomposing every implication in the policy using the rule ($\to$L) in every branch, which takes time exponential in n because the rule ($\to$L) has two premises. As explained

in the next section, goal-directed search tries to prove the antecedents of only those implications that can help prove the goal. Thus it first tries to prove p_1, then p_2, etc. This results in an almost linear search space (the actual curve is quadratic because at each point in this search space, the procedure must try all n policy assumptions; however, all but one are immediately pruned). This simple but realistic example illustrates that goal-directed search can indeed be a pragmatic choice for theorem proving with access policies and motivates our technical work.

3 $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$: Goal-directed Access Control Logic

Our main technical contribution is the development of the Hereditary-Harrop or HH fragment of $\mathrm{BL}_{\mathrm{sf}}$, which we call $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, and a goal-directed proof search procedure for it, along with associated soundness, completeness and termination proofs. The HH fragment of first-order logic [17] is a generalization of the Horn fragment on which Prolog works, but still admits Prolog's top-down proof search. Our work generalizes this fragment to also include the connectives A says φ and A sf B and our proof search marries the formalism of backchaining in Prolog with labeled calculi, which we believe to be a novel contribution. The syntax of formulas in the fragment $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ is stratified into three categories.

$$
\begin{array}{ll}
\text{Goals} & G ::= p \mid A \text{ says } G \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid N \rightarrow G \mid \top \mid \bot \\
\text{Clauses} & D ::= p \mid G \rightarrow D \mid D_1 \wedge D_2 \mid \top \mid \bot \mid A \text{ says } D \\
\text{Chunks} & N ::= D \mid N_1 \vee N_2 \mid N_1 \wedge N_2 \mid A \text{ sf } B
\end{array}
$$

In our goal-directed calculus, goals can appear only on the right side in sequents, whereas clauses and chunks appear only on the left side in separate contexts. In logic programming notation, one may think of goals as the allowed queries, clauses as rules that constitute logic programs and chunks as additional constructs that combine logic programs (note that the leaf of any chunk is either a clause or A sf B). Not every connective is allowed in every category of syntax; this is necessary to guarantee completeness of goal-directed search. We do not go into the details of why this is the case as it is a standard but technically deep result in proof theory. The new interesting innovation here is deciding where to allow and disallow the new connectives A says φ and A sf B. (Interested readers are referred to existing work for details of the restriction, specifically [17] and [9, Chapter 6].)

Even though $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ is less expressive than $\mathrm{BL}_{\mathrm{sf}}$ it is still much more expressive than Datalog-based access control languages like SecPAL[6] or DKAL[15], which also have efficient decision procedures. The main difference between a Datalog-based language and $\mathrm{BL}_{\mathrm{sf}}$ is that the former will disallow disjunction altogether and also disallow implication in goals but both connectives can be useful in some cases (our next example illustrates this point). In return, Datalog-based languages have worst-case polynomial time decision procedures, which we forgo, settling for a bad worst-case execution time but efficient execution on policies of interest.

Example 1 ($\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ *Expressiveness).* Suppose that *Alice* wants to share a picture $pic1$ with the following policy (where $\mathcal{C}$ represents the finite domain of Alice's contacts): "Members of the group family can access $pic1$ if *none* of them is a friend of a colleague of mine". Such a policy can be expressed in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ as follows:

$$Alice \text{ says } \Big(\bigwedge_{x \in \mathcal{C}} (family\_of(x, Alice) \to \bigwedge_{y \in \mathcal{C}} (colleague\_of(y, Alice) \to \neg friend\_of(y, x))\Big) \\ \to \Big(\bigwedge_{x \in \mathcal{C}} (family\_of(x, Alice) \to can\_access\_x\_pic1)\Big)$$

This example cannot be expressed in any Datalog-based access control logic because it uses left-nested implication.

3.1 SeqG: Goal-directed proof theory for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$

We now describe a goal-directed proof calculus, SeqG, for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$. Following standard literature, the calculus uses more that one type of sequent; we describe each of them. A key difference from SeqC is that we allow only one formula in the right side of a sequent. This is complete because we are working in an intuitionistic logic, which has a disjunction property. Roughly, $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ implies that there is some labeled formula $x : \varphi \in \Delta$ such that $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \varphi$. (We note that this is only an intuition. The precise disjunction property we need is presented in our technical report.)

It is simplest to think of the inference rules of SeqG as describing a proof search method, obtained by reading the inference rules bottom-up. Proof search starts in an *R-sequent*, which has the form $\Sigma; \mathbb{M}; \Gamma \Mapsto x : G$. Here Γ is a multiset of labeled clauses of the form $w : D$ (no chunks are allowed in Γ in SeqG). More importantly, the rules for inferring a given R-sequent are *deterministic*: For each possible value of G, there is one rule that decomposes G into its subformulas in the premises, following the corresponding right rule in SeqC (see Figure 3). This forced decomposition of goal formulas justifies the adjective "goal-directed" for our calculus. The only exception to goal-directed decomposition is that after decomposing a goal of the form $N \to G'$, we push the chunk N into a special context denoted Ξ on the left and transition into what we call an *L-sequent*, where N is immediately decomposed with left rules (described below). After N has been decomposed completely, we return to work on G'. Eventually, in each branch, G must decompose to $\top$ (which succeeds immediately), $\bot$ (which fails immediately) or an atom p. In the last case, we backchain (through *N-sequents*), as in Prolog, by picking a suitable clause in Γ to prove p.

An L-sequent has the form $\Sigma; \mathbb{M}; \Gamma; \Xi \Leftarrow x : G$, where Ξ is a set of labeled chunks of the form $w : N$. The inference rules for an L-sequent (Figure 3) decompose the formulas in Ξ with left rules from SeqC. The results are of the form $w : D$, which are pushed into Γ (rule (pr)). Once Ξ is empty, search transitions to an R-sequent (rule (L2R)).

When a goal is reduced to an atom, search transitions from an R-sequent to an N-sequent of the form $\Sigma; \mathbb{M}; \Gamma \Leftrightarrow w : p$ (rule (atom)). There is only one rule to prove an N-sequent. This rule, called (choice), is the site of backchaining:

It non-deterministically picks a clause $x : D$ from Γ (first premise), uses an *F-sequent* to determine what additional subgoals $w_1 : G_1, \ldots, w_n : G_n$, when proved, will cause $x : D$ to imply $w : p$ (second premise) and then tries to prove the subgoals (third premise).

F-sequents have the form $\Sigma; \mathbb{M}; x : D \ll w : p \mid Gl$ (here, $\ll$ replaces the sequent arrow; it is not a binary operator). The meaning of the sequent is that if all (labeled) subgoals in list Gl hold, then $x : D$ entails $w : p$. In an implementation, Gl is an output. Rules for proving F-sequents are mostly deterministic and work by decomposing D. The only exception to this determinism is the choice of rules ($\wedge L_1$) and ($\wedge L_2$) when $D = D_1 \wedge D_2$. Note that if no head in D matches p, then $\Sigma; \mathbb{M}; x : D \ll w : p \mid Gl$ cannot be established for any Gl.

There are two key points to observe about the calculus SeqG. First, unlike SeqC, there are no frame rules. Instead, the effect of all frame rules is collected into the operator $\overline{\Sigma; \mathbb{M}; \Gamma}$ in the premise of the rule (atom). This operator informally means "apply frame rules of SeqC to $\Sigma; \mathbb{M}; \Gamma$ to the extent possible". (For a formal definition, see our technical report.) Second, SeqG is largely a deterministic calculus. The *only* real source of non-determinism is in picking a clause in the first premise of the rule (choice) and in choosing between rules ($\wedge L_1$) and ($\wedge L_2$) in an F-sequent. It is because of this highly deterministic nature that proof search in SeqG works very efficiently in practice.

4 BL^{G}_{sf}: Soundness, Completeness and Termination

We prove formally that for the fragment BL^{G}_{sf}, the goal-directed calculus SeqG is sound and complete with respect to the calculus SeqC (and, hence, by Theorem 4, also with respect to the Kripke semantics). The proof of completeness (if direction of the theorem) is particularly non-trivial. We build on an earlier proof [9] for a variant of the logic BL_{sf}, but in the non-labeled setting.

Theorem 5 (Soundness and Completeness) $\vdash (\Sigma; \mathbb{M}; \Gamma \Mapsto x : \varphi)$ *in SeqG if and only if* $\vdash (\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \varphi)$ *in SeqC.*

Next, we prove that SeqG can be used as a decision procedure, by bounding the maximum *depth* up to which branches need to be searched. Our argument is based on a reduction to SeqC. We first show that if a sequent has a proof of depth k in SeqC, it has a proof of depth no more than $3k+1$ in SeqG. Second, we show that for every sequent, we can compute a number n such that if the sequent is provable, then it has a proof of depth n in SeqC. Together, the two imply that we can prune search in SeqG at depth $3n + 1$ without losing completeness. The first fact follows by a careful analysis of the proof of completeness in Theorem 5. As far as we are aware, this is a novel result for goal-directed search. The second fact follows from a combinatorial analysis of the termination proof of EW.

Lemma 6 *If* $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \varphi$ *is provable in SeqC with a derivation of depth* k *then* $\Sigma; \mathbb{M}; \Gamma \Mapsto x : \varphi$ *is provable in SeqG with a derivation of depth at most* $3k + 1$.

R sequents

$$\frac{}{\Sigma;\mathbb{M};\Gamma \Leftrightarrow x:p}\,\text{atom} \qquad \frac{\Sigma;\mathbb{M};\Gamma \Rightarrow x:G_i}{\Sigma;\mathbb{M};\Gamma \Rightarrow x:G_1 \vee G_2}\vee\text{R}_i \qquad \frac{\Sigma,y;\mathbb{M},x\leq y;\Gamma;y:N \Leftarrow y:G}{\Sigma;\mathbb{M};\Gamma \Rightarrow x:N \to G}\to\text{R}$$

L sequents

$$\frac{\Sigma;\mathbb{M};\Gamma \Rightarrow w:G}{\Sigma;\mathbb{M};\Gamma;\cdot \Leftarrow w:G}\text{L2R} \qquad \frac{\Sigma;\mathbb{M};\Gamma,x:D;\Xi \Leftarrow w:G}{\Sigma;\mathbb{M};\Gamma;\Xi,x:D \Leftarrow w:G}\text{pr} \qquad \frac{\Sigma;\mathbb{M};\Gamma;\Xi,x:N_1,x:N_2 \Leftarrow w:G}{\Sigma;\mathbb{M};\Gamma;\Xi,x:N_1\wedge N_2 \Leftarrow w:G}\wedge\text{L}$$

$$\frac{}{\Sigma;\mathbb{M};\Gamma;\Xi,x:\bot \Leftarrow w:G}\bot\text{L} \qquad \frac{\Sigma;\mathbb{M};\Gamma;\Xi,x:N_1 \Leftarrow w:G \qquad \Sigma;\mathbb{M};\Gamma;\Xi,x:N_2 \Leftarrow w:G}{\Sigma;\mathbb{M};\Gamma;\Xi,x:N_1\vee N_2 \Leftarrow w:G}\vee\text{L}$$

N sequents

$$\frac{x:D\in\Gamma \qquad \Sigma;\mathbb{M};x:D \ll w:p \mid w_1:G_1 \ldots w_n:G_n \qquad (\Sigma;\mathbb{M};\Gamma \Rightarrow w_i:G_i)_{i=1}^{n}}{\Sigma;\mathbb{M};\Gamma \Leftrightarrow w:p}\,\text{choice}$$

F sequents

$$\frac{}{\Sigma;\mathbb{M},x\leq w;x:q \ll w:p \mid \bullet}\,\text{init} \qquad \frac{}{\Sigma;\mathbb{M};x:A\ \mathsf{sf}\ B \ll x:A\ \mathsf{sf}\ B \mid \bullet}\,\text{sf}$$

$$\frac{\Sigma;\mathbb{M};x:D_i \ll w:p \mid Gl}{\Sigma;\mathbb{M};x:D_1\wedge D_2 \ll w:p \mid Gl}\wedge\text{L}_i \qquad \frac{x\leq y\in\mathbb{M} \qquad \Sigma;\mathbb{M};y:D \ll w:p \mid Gl}{\Sigma;\mathbb{M};x:G\to D \ll w:p \mid y:G,Gl}\to\text{L}$$

$$\frac{xS_Ay\in\mathbb{M} \qquad \Sigma;\mathbb{M};y:D \ll w:p \mid Gl}{\Sigma;\mathbb{M};x:A\ \mathsf{says}\ D \ll w:p \mid Gl}\,\mathsf{says}\text{L}$$

Fig. 3. SeqG: A goal-directed calculus for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$, selected rules

Lemma 7 *For every sequent $\Sigma;\mathbb{M};\Gamma \Rightarrow x:\varphi$ a number n can be computed such that if $\Sigma;\mathbb{M};\Gamma \Rightarrow x:\varphi$ has a proof in SeqC, then it has a proof of depth less than n.*

Corollary 8 (Termination) *The rules of SeqG can be used as a decision procedure for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ by pruning search in each branch at depth $3n+1$ where n is the number from Lemma 7 for the starting sequent.*

5 SeqG$^{\mathrm{AB}}$: Policy Abduction in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$

We now extend the goal-directed sequent calculus SeqG to an abduction procedure. An abduction procedure, upon failure to find a proof, emits possible ways to extend the theory to complete the proof. In the context of access control, this amounts to listing additional credentials that will authorize a desired access. Our abduction calculus is directly based on a similar calculus from EW (that calculus extends SeqC). Although not a significant technical innovation, our abduction procedure works much faster in practice than that of EW, mirroring the difference between SeqC and SeqG (Section 2).

R sequents

$$\frac{}{\Sigma;\mathbb{M};\Gamma \Leftrightarrow x : p \searrow_{m-1} \Theta}\text{atom}$$

$$\frac{\Sigma;\mathbb{M};\Gamma \Rightarrow x : G_1 \searrow_{m-1} \Theta_1 \qquad \Sigma;\mathbb{M};\Gamma \Rightarrow x : G_2 \searrow_{m-1} \Theta_2}{\Sigma;\mathbb{M};\Gamma \Rightarrow x : G_1 \vee G_2 \searrow_m \Theta_1 \vee \Theta_2}\vee\text{R}$$

$$\frac{\Sigma, y;\mathbb{M}, x \leq y;\Gamma; y : N \Leftarrow y : G \searrow_{m-1} \Theta}{\Sigma;\mathbb{M};\Gamma \Rightarrow x : N \to G \searrow_m \Theta}\to\text{R}$$

N sequents

$$\frac{T = \bigvee_{x:D\in\Gamma} \left\{ (\Theta_1 \wedge \ldots \wedge \Theta_n) \;\middle|\; \begin{array}{l} \Sigma;\mathbb{M}, x : D \ll w : p \mid Gl \text{ and} \\ (Gl = g_1, \ldots, g_n) \text{ and} \\ (\Sigma;\mathbb{M};\Gamma \Rightarrow g_i \searrow_{m-1} \Theta_i) \end{array} \right\}}{\Sigma;\mathbb{M};\Gamma \Leftrightarrow w : p \searrow_m T \vee AB(\Sigma;\mathbb{M};\Gamma; w : p)}\text{choice}$$

$$\frac{\text{No other rule applicable or } (m-1=0)}{\Sigma;\mathbb{M};\Gamma \Leftrightarrow w : p \searrow_m AB(\Sigma;\mathbb{M};\Gamma; w : p)}\text{AB}$$

Fig. 4. SeqG$^{\text{AB}}$: Abducible extraction for BL$^{\mathsf{G}}_{\text{sf}}$, selected rules

Our abduction procedure is also presented as a calculus, SeqG$^{\text{AB}}$, shown in Figure 4. Judgments of the calculus have the form $\Sigma;\mathbb{M};\Gamma \Rightarrow z : \varphi \searrow_m \Theta$. Here, m is a counter that decrements with each backward application of a rule. Initially, it is set to a value greater than the bound $3n+1$ of Corollary 8. When m reaches 0 or when no other rule of SeqG can be applied, we apply rule (AB) to output an abducible Θ, which is a representation of additional credentials that could be used to prove the goal. Formally, Θ is a just a logical formula containing atoms p and formulas A says p at the leaves and only the connectives $\top$, $\bot$, $\vee$ and $\wedge$. The function $\text{AB}(\Sigma;\mathbb{M};\Gamma; w : p)$, defined in EW, extracts such an abducible from the N-sequent. We reproduce the definition here:

$$\text{AB}(\Sigma;\mathbb{M};\Gamma; w : p) = \begin{pmatrix} (\bigvee\{p \mid (\text{root}(\mathbb{M})) \leq w \in \mathbb{M}\}) \vee \\ (\bigvee\{A \text{ says } p \mid (\text{root}(\mathbb{M}))S_A w \in \mathbb{M}\}) \end{pmatrix}$$

Here, root($\mathbb{M}$) is the world of $\mathbb{M}$ with which the first goal formula was labeled. Intuitively, for a given goal $w : p$, we look at the path between the root of $\mathbb{M}$ and y. Because the $\Sigma, \mathbb{M}$ and Γ are closed under backward application of rules (I), (mon-S) and (trans) (due to the closure in the premise of the rule (atom)), either $(\text{root}(\mathbb{M})) \leq y \in \mathbb{M}$ or $(\text{root}(\mathbb{M}))S_A y \in \mathbb{M}$ for some $A \in \mathcal{I}$. In the former case, it suffices to add the credential p to complete the proof and in the latter case it suffices to add the credential A says p to complete the proof. If both sets in the definition of $\text{AB}(\Sigma;\mathbb{M};\Gamma; w : p)$ are empty, then $\text{AB}(\Sigma;\mathbb{M};\Gamma; w : p) = \bot$.

An abducible Θ is *satisfied* by extending the current policy Γ with a set $F \subseteq \{p, A \text{ says } p \mid A \in \mathcal{I}\}$. Given such a set, we define the satisfaction relation $F \models \Theta$ in the obvious way: $F \models p$ iff $p \in F$; $F \models A$ says p iff $(A \text{ says } p) \in F$; $F \models \Theta_1 \wedge \Theta_2$ iff $F \models \Theta_1$ and $F \models \Theta_2$; etc. The following theorem states that

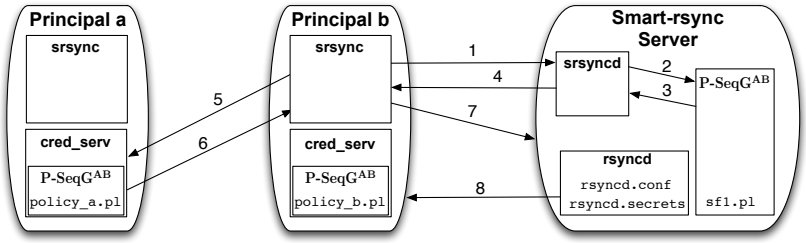

Fig. 5. The Smart-rsync Architecture

our abduction procedure is sound in the sense that if the abducible of a sequent is satisfied by F, then extending the hypotheses with F results in a provable sequent.

Theorem 9 (Soundness). *If* $\Sigma; \mathbb{M}; \Gamma \Rightarrow w : p \searrow_m \Theta$ *and* $F \models \Theta$, *then* $\Sigma; \mathbb{M}; \Gamma \cup root(\mathbb{M}) : F \Rightarrow w : p$.

A Prolog Implementation We have implemented both the goal-directed calculus SeqG and its extension with abducibles, SeqG$^{\mathrm{AB}}$, in Prolog and tested it with Prolog's two major interpreters, SWI-Prolog and GNUProlog. Our implementation closely mirrors the description of Sections 3.1 and 5. The implementation was used to test performance (e.g., Figure 2) and also as a black-box interpretation engine in a larger case-study with `rsync` that is described next.

6 Smart-rsync: Distributed File Synchronization with SeqG$^{\mathrm{AB}}$

As a case study in the use of $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ and SeqG$^{\mathrm{AB}}$, we extend the Unix synchronization program `rsync` to use $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ for representing authorization policies and SeqG$^{\mathrm{AB}}$ to automatically gather credentials needed for authorization. The implementation consists of three new programs: `srsyncd`, `srsync` and `cred_serv` that run on different servers and clients. We also use the standard `rsync` server daemon, `rsyncd`, unmodified and a Prolog implementation of SeqG$^{\mathrm{AB}}$. Figure 5 illustrates the various components of our implementation and their interaction with each other.

A client (principal b in the figure) wanting to synchronize a local file f with the server's version calls the server with our client program `srsync` (step 1 in the figure). This call is received by the server's daemon `srsyncd`. The daemon authenticates the request using a signed certificate accompanying the request, then it invokes an implementation of the calculus SeqG$^{\mathrm{AB}}$ on its policy represented in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ to determine what additional credentials, if any, are needed to authorize the access (step 2). The policy is provided to the server by an administrator through a configuration file. If no additional credentials are required to

authorize the request, the server adds the permission to file f for principal b to a local permission file, `rsyncd.secrets`, and informs the client program of the update. The client program `srsync` then jumps to step 7 (described below) to synchronize the file.

If additional credentials are required, SeqG$^{\mathrm{AB}}$ returns to `srsyncd` an abducible (step 3), which `srsyncd` returns to the client (step 4). The client program, `srsync`, then communicates with the credential servers, `cred_server`, on various remote hosts to obtain the required credentials to satisfy the abducible. In general, to obtain a credential of the form A says p, it contacts the credential server of principal A (the principal to IP address mapping is provided to `srsync` in a configuration file). If there is more than one way to satisfy the abducible (because of the connective $\vee$ in the abducible), `srsync` tries the alternatives one at a time. In the figure, principal b contacts principal a for some credential (step 5). The contacted credential server then checks its own policy, also written in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$, via a call to SeqG$^{\mathrm{AB}}$, to find further abducibles needed to return the requested credentials. The client then recursively satisfies those abducibles. Eventually, this recursive process ends and the client is returned the requested credential (step 6), which it then passes to `srsyncd` to complete its authorization (step 7). At this point, `srsyncd` sets the permission to file f for the client's IP address in a special file `rsyncd.secrets`. The client then invokes the standard Unix program `rsyncd` on the server, which runs in parallel with `srsyncd`, to complete the synchronization (steps 7 and 8). `rsyncd` is configured to read permissions from `rsyncd.secrets`.

In our limited experience with Smart-rsync, we found it relatively easy to use. This is mainly given by two reasons. First, the proof construction process, including the recursive generation of abducibles, is totally automatic and, therefore, transparent to the user. Effort is involved in only setting the right policies, but this cannot be avoided in any case. Second, SeqG$^{\mathrm{AB}}$ works very efficiently (Section 2), so there isn't much visible overhead.

7 Related Work

Although $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is a fragment of $\mathrm{BL}_{\mathrm{sf}}$, its goal-directed proof theory is intrinsically different from the one presented in EW for $\mathrm{BL}_{\mathrm{sf}}$ [14]. Whereas the proof theory of $\mathrm{BL}_{\mathrm{sf}}$, SeqC, is a standard labeled sequent calculus, the proof theory of $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$, SeqG, is a marriage of labeled sequent calculi with backchaining search. SeqG works much faster in practice than SeqC but, unlike SeqC, SeqG cannot be used for policy saturation and countermodel generation.

There has also been prior work on goal-directed search in a related logic BL, presented in the second author's thesis [9, Chapter 6], but that work is based in an unlabeled sequent calculus. It does not consider the formula A sf B, but it does consider general first-order quantifiers. Without the use of labels, it is necessary to limit the nesting depth of A says φ in policies to 1 in the Hereditary Harrop fragment, so a policy like A says B says φ cannot be expressed in the goal-directed fragment of [9] (but can be expressed in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$). Our proof of completeness of goal-directed search uses the same structure as that of [9].

Besides $\mathrm{BL}_{\mathrm{sf}}$ and BL, there is also a significant amount of work on goal-directed search in Datalog-based authorization languages. For example, the trust management language Soutei [19] is an extension of Datalog with domains that are similar to the connective A says φ and its implementation uses distributed backchaining search. The authorization policy language SecPAL [6], based on Datalog with constraints, uses tabled backchaining search to decide access policies in polynomial time. However, as discussed in Section 3, Datalog is less expressive than $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$.

8 Conclusion

We have presented $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, a goal-directed fragment of $\mathrm{BL}_{\mathrm{sf}}$ [14], and developed SeqG, a sound, complete and terminating goal-directed proof search based on it. We have explained through examples and simple experiments that although of the same worst-case complexity as $\mathrm{BL}_{\mathrm{sf}}$, $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ works much faster on realistic authorization policies. We have also modified SeqG to obtain an abduction calculus that produces missing credentials when an authorization fails and implemented it for performing automatic discovery of missing credentials in an extension of the Unix file synchronization program `rsync`.

The design space of access control logics is wide, ranging from very expressive, but intractable higher-order logics to restrictive, but efficiently decidable Datalog-based logics. We believe that goal-directed search over the Hereditary Harrop fragment, as in $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, represents a reasonable balance of expressiveness and practical tractability.

References

1. Abadi, M.: Logic in access control. In: Proceedings of the IEEE Symposium on Logic in Computer Science (LICS). pp. 228–233 (2003)
2. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A calculus for access control in distributed systems. ACM Transactions on Programming Languages and Systems (TOPLAS) 15(4), 706–734 (1993)
3. Avijit, K., Datta, A., Harper, R.: Distributed programming with distributed authorization. In: Proceedings of the ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI). pp. 27–38 (2010)
4. Bauer, L.: Access Control for the Web via Proof-Carrying Authorization. Ph.D. thesis, Princeton University (2003)
5. Bauer, L., Garriss, S., McCune, J.M., Reiter, M.K., Rouse, J., Rutenbar, P.: Device-enabled authorization in the Grey system. In: Proceedings of the International Conference on Information Security (ISC). pp. 431–445 (2005)
6. Becker, M.Y., Fournet, C., Gordon, A.D.: SecPAL: Design and semantics of a decentralized authorization language. Journal of Computer Security 18(4), 619–665 (2010)
7. Becker, M.Y., Russo, A., Sultana, N.: Foundation of logic-based trust management. In: Proceedings of the IEEE Symposium on Security and Privacy. pp. 161–175 (2012)

8. DeTreville, J.: Binder, a logic-based security language. In: Proceedings of the IEEE Symposium on Security and Privacy. pp. 105–113 (2002)
9. Garg, D.: Proof Theory for Authorization Logic and Its Application to a Practical File System. Ph.D. thesis, Carnegie Mellon University (2009)
10. Garg, D., Pfenning, F.: Non-interference in constructive authorization logic. In: Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW). pp. 283–293 (2006)
11. Garg, D., Pfenning, F.: A proof-carrying file system. In: Proceedings of the IEEE Symposium on Security and Privacy. pp. 349–364 (2010)
12. Genovese, V., Rispoli, D., Gabbay, D.M., van der Torre, L.: Modal access control logic: Axiomatization, semantics and FOL theorem proving. In: Proceedings of the Fifth Starting AI Researchers' Symposium (STAIRS). pp. 114–126 (2010)
13. Genovese, V., Garg, D., Rispoli, D.: Labeled goal-directed search in access control logic (2012), Technical Report. Online at http://www.mpi-sws.org/~dg/
14. Genovese, V., Garg, D., Rispoli, D.: Labeled sequent calculi for access control logics: Countermodels, saturation and abduction. In: Proceedings of the 25th IEEE Symposium on Computer Security Foundations (CSF). pp. 139–153 (2012)
15. Gurevich, Y., Neeman, I.: Logic of infons: The propositional case. ACM Transactions on Programming Languages and Systems (TOPLAS) 12(2) (2011)
16. Jia, L., Vaughan, J.A., Mazurak, K., Zhao, J., Zarko, L., Schorr, J., Zdancewic, S.: Aura: A programming language for authorization and audit. In: Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 27–38 (2008)
17. Miller, D., Nadathur, G., Pfenning, F., Scedrov, A.: Uniform proofs as a foundation for logic programming. Annals of Pure and Applied Logic 51, 125–157 (1991)
18. Negri, S.: Proof analysis in modal logic. Journal of Philosophical Logic 34, 507–544 (2005)
19. Pimlott, A., Kiselyov, O.: Soutei, a logic-based trust-management system. In: Proceedings of the 8th International Conference on Functional and Logic Programming (FLOPS). pp. 130–145 (2006)
20. Schneider, F.B., Walsh, K., Sirer, E.G.: Nexus Authorization Logic (NAL): Design rationale and applications. ACM Transactions on Information and System Security (TISSEC) 14(1), 1–28 (2011)
21. Swamy, N., Chen, J., Fournet, C., Strub, P.Y., Bhargavan, K., Yang, J.: Secure distributed programming with value-dependent types. In: Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 266–278 (2011)
22. Viganò, L.: A framework for non-classical logics. Ph.D. thesis, Universität des Saarlandes (1997), also available as the book *Labelled non-classical logics*, Springer 2000.