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 it finds proofs of authorization quickly in practice. We also extend $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$'s proof 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`.

1 Introduction

Many access control systems rely on representation of authorization policies as logical theories [4–6, 13, 17, 22]. 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 [5, 6, 13], as well as formal proofs of policy meta properties like non-interference [8, 12]. 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 [3, 9, 12–14, 21]. 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 recent work, Genovese, Garg and Rispoli (called GGR in the sequel) [15] prove that an expressive access control logic, $\mathrm{BL}_{\mathrm{sf}}$, is decidable and use its decision procedure as a foundation to solve three other practical problems in access

control (in addition to theorem proving): 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 a nice theoretical result, experimental evaluation of the GGR 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 wide applicability of the decision procedure, the question we ask is whether or not that complexity can be reduced in practice, perhaps at the cost of sacrificing some of its 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* [18] 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 GGR, our procedure is based in labeled sequent calculi [19, 23], 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 GGR 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 GGR.[1]

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

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 labaled 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 GGR, the method of labeled sequent calculi and GGR's decision procedure, and explain by experimental evaluation the exponential behavior of the GGR 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 [1].

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 GGR [15]. $\mathrm{BL}_{\mathrm{sf}}$ 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 $\to$ (implication) have usual meanings from intuitionistic logic.

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

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

$$\frac{\text{(All intuitionistic propositional tautologies)}}{\vdash \varphi} \qquad \frac{\vdash \varphi}{\vdash A \text{ says } \varphi} \text{ (nec)}$$

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

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

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

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

Rule (nec) and axiom (K) are standard in modal logic. Axiom (I) is neeeded to accurately model delegation in the logic [2], whereas (speaksfor) characterizes the formula $A \mathsf{sf} B$: If $A \mathsf{sf} B$, then any statement φ that A makes is echoed by B, so the formula $A \mathsf{sf} B$ means that A has authority to speak on behalf of B [3].

Kripke Semantics The semantics of $\mathrm{BL}_{\mathrm{sf}}$ are presented in the standard, Kripke style for modal logics. A model of the logic 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 $(A \mathsf{says} \cdot)$. Intuitively, if wS_Aw' then principal A believes that world w' is a potential (knowledge) successor of the world w. Intuitionistic implication is modeled using a binary preorder, $\leq$. GGR treat the formula $A \mathsf{sf} B$ as an atom in the Kripke semantics and validate axioms related to it, e.g., (speaksfor), through conditions on Kripke frames. This interpretation is very distinct from earlier interpretations of $A \mathsf{sf} B$, e.g., [3, 11], that define $A \mathsf{sf} B$ in terms of relations between accessibility relations S_A and S_B, but this choice simplifies the decision procedure.

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 $A \mathsf{sf} B$ holds.*

Let $S_* = \bigcup_{A \in \mathcal{I}} S_A$. 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)*
- $\forall x, y, z.(((x \leq y) \wedge (yS_Az)) \to (xS_Az))$ *(mon-S)*
- $\forall x, y, z.(((xS_By) \wedge (yS_Az)) \to (xS_Az))$ *(I)*
- *If $w \in sf(A, B)$, then for all w', wS_Bw' implies wS_Aw'.* *(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 $x(\leq \cup S_*)^* y$ 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 $A \mathsf{sf} B$ 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 $A \mathsf{sf} B$ 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 GGR, on which the entire technical development of GGR 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 GGR's sequent calculus, so we discuss the sequent calculus is some detail here. Following standard presentations of labeled sequent calculi (also called prefixed calculi), GGR's calculus, SeqC, manipulates two types of labeled formulas: *world formulas*, written $w : \varphi$, 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 GGR 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 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 [15]). $\vdash (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$ *has a proof if and only if* $\models (\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta)$

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

$$\frac{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \alpha, x : \alpha \wedge \beta, \Delta \qquad \Sigma; \mathbb{M}; \Gamma \Rightarrow x : \beta, x : \alpha \wedge \beta, \Delta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \alpha \wedge \beta, \Delta}\wedge\text{R}$$

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

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

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

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: GGR's labeled sequent calculus for BL_{sf}, selected rules

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 GGR 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 BL_{sf}. (Further, GGR describes how counter-models 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 GGR'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. GGR do not observe this fact, but it is not difficult to prove by a careful analysis of their termination proof. We use this observation to derive a similar completeness-preserving bound ($3n + 1$ to be exact) for our goal-directed search procedure, thus implying that our procedure can also be converted to a decision procedure.

Second, the number n is large – it is doubly exponential in the size of the sequent. Hence, the worst-case complexity of both our GGR's decision procedure for $\mathrm{BL}_{\mathrm{sf}}$ and ours for $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is also doubly exponential. The difference, and this is the key point, is that on *practical policies*, GGR'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 that Prolog works efficiently in practice; here, we merely exploit this known fact for access control.

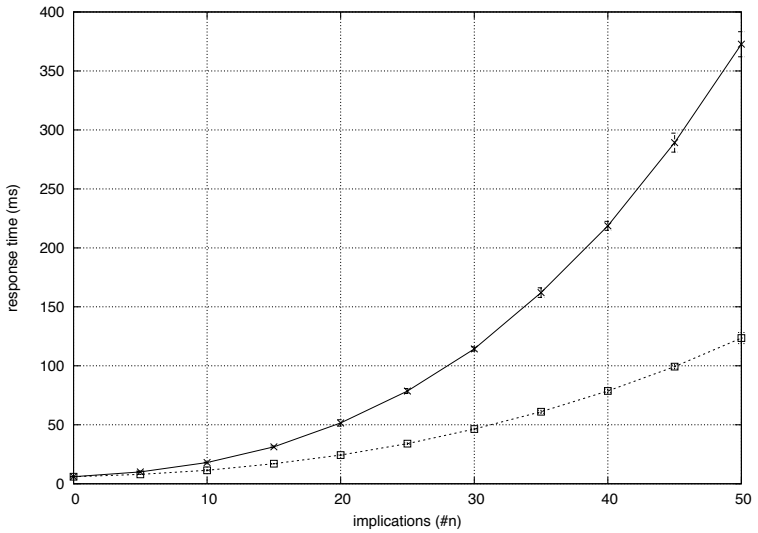

Fig. 2. Scalability of decision procedures based on SeqC and SeqG. The solid line is SeqC and the dotted line is SeqG.

To illustrate the second point, we compare the time taken by an implementation of our decision procedure to that taken by GGR's decision procedure on a simple, common policy: $\Gamma_n = \{p_1 \to sf \wedge p_2 \to p_1 \wedge \ldots \wedge p_n \to p_{n-1}\}$. Here, $p_1, \ldots, p_n$ and sf are atoms. The goal is to test the unprovable fact $\Gamma_n \Rightarrow sf$. This example is representative how real access policies work, e.g., sf 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 GGR's procedure and our goal-directed procedures for this example are shown in Figure 2. The upper solid line, corresponding to GGR, 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: GGR'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 $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$: 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 $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$, and the development of a goal-directed proof search procedure for it, along with associated soundness, completeness and termination proofs. The HH fragment of first-order logic [18] 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 search in Prolog with labeled calculi, which we believe to be a novel contribution. The syntax of formulas in the fragment $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is stratified into three categories.

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

In our goal-directed sequent calculus goals can only appear 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 refered to existing work for details of the restriction, specifically [18].)

Even though $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is less expressive than $\mathrm{BL}_{\mathrm{sf}}$ it is still much more expressive than Datalog-based access control languages like SecPAL[7] or DKAL[16], 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 forego, settling for a bad worse-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 *pic*1 with the following policy (where $\mathcal{C}$ represents the finite domain of contacts): "Members of the group family can access *pic*1 if *none* of them is a friend of a colleague of mine". Such policy can be expressed in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ as follows:

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

This example cannot be expressed in any Datalog-based access control logic.

Because $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is defined as a syntactic restriction of $\mathrm{BL}_{\mathrm{sf}}$ the meaning of $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ connectives is formally defined as in Section 2. From the proof-theoretic point of view, an important difference between $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ and $\mathrm{BL}_{\mathrm{sf}}$ is that $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ enjoys the disjunction property which is pivotal in order to be able to develop a goal-directed proof theory. In what follows we prove that, if we constrain the language of $\mathrm{BL}_{\mathrm{sf}}$ as illustrated above, the calculus SeqC as reported in Figure 1 enjoys the disjunction property.

Definition 5 *Given a sequent* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ *and a world* $w \in \Sigma$ *we define the* restriction of contexts to ancestors *of* w *as* $\Sigma_w; \mathbb{M}_w; \Gamma_w \Rightarrow \Delta_w$ *as follows*

$$\Sigma_w = \{x \in \Sigma \mid \exists x_1, \ldots, x_n \ s.t. \ x \circ x_1 \circ \ldots \circ x_n \circ w\} \ with \circ \in \{S_*, \leq\}$$
$$\mathbb{M}_w = \{x \circ y \in \mathbb{M} \mid x, y \in \Sigma_w\} \ with \circ \in \{S_*, \leq\}$$
$$\Gamma_w = \{x : \varphi \in \Gamma \mid x \in \Sigma_w\}$$

Theorem 6 (Height-preserving Disjunction Property) *If disjunction does not appear in negative positions and if* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ *is derivable with a proof of height* h*, then there is some* $w : \varphi \in \Delta$ *such that* $\Sigma_w; \mathbb{M}_w; \Gamma_w \Rightarrow w : \varphi$ *is derivable with a proof of at most height* h*.*

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 $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \varphi$. (See Theorem 6).

It is simplest to think of the inference rules of SeqG as describing a proof search method, obtained by reading the rules bottom-up. Proof search starts in an *R-sequent*, which has the form $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G$. 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 infering 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 $\Sigma; \mathbb{M}; x : D \ll w : p \mid Gl$. The meaning of the sequent is that if all (labeled) subgoals in 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 there is no head in D that matches p, there is no way to establish $\Sigma; \mathbb{M}; x : D \ll w : p \mid Gl$ 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 Definition 7) 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.

This choice of picking the clause is the *only* point of non-determinism in backwards search in SeqG, which is why proof search in it works very efficiently in practice.

Definition 7 (Saturation with respect to Frame Rules) *The* $\overline{\Sigma;\mathbb{M};\Gamma}$ *in rule (atom) represents the saturation w.r.t. the semantic conditions reported in Definition 1 and it is defined as the fixed-point of what follows:*

1. $\overline{\Sigma, x;\mathbb{M};\Gamma} = \overline{\Sigma, x;\mathbb{M};\Gamma, x : A \;\mathsf{sf}\; A}$
2. $\overline{\Sigma;\mathbb{M}, xS_By;\Gamma, x : A \;\mathsf{sf}\; B} = \overline{\Sigma;\mathbb{M}, xS_By, xS_Ay;\Gamma, x : A \;\mathsf{sf}\; B}$
3. $\overline{\Sigma;\mathbb{M};\Gamma, x : A \;\mathsf{sf}\; B, x : B \;\mathsf{sf}\; C} = \overline{\Sigma;\mathbb{M};\Gamma, x : A \;\mathsf{sf}\; B, x : B \;\mathsf{sf}\; C, x : A \;\mathsf{sf}\; C}$
4. $\overline{\Sigma;\mathbb{M}, xS_Cy;\Gamma, x : A \;\mathsf{sf}\; B} = \overline{\Sigma;\mathbb{M}, xS_Cy;\Gamma, x : A \;\mathsf{sf}\; B, y : A \;\mathsf{sf}\; B}$
5. $\overline{\Sigma;\mathbb{M}, x \leq y;\Gamma, x : A \;\mathsf{sf}\; B} = \overline{\Sigma;\mathbb{M}, x \leq y;\Gamma, x : A \;\mathsf{sf}\; B, y : A \;\mathsf{sf}\; B}$
6. $\overline{\Sigma;\mathbb{M}, x \leq y, yS_Az, z \leq w;\Gamma} = \overline{\Sigma;\mathbb{M}, x \leq y, yS_Az, z \leq w, xS_Aw;\Gamma}$
7. $\overline{\Sigma, x;\mathbb{M};\Gamma} = \overline{\Sigma, x;\mathbb{M}, x \leq x;\Gamma}$
8. $\overline{\Sigma;\mathbb{M}, x \leq y, y \leq z;\Gamma} = \overline{\Sigma;\mathbb{M}, x \leq y, y \leq z, x \leq z;\Gamma}$
9. $\overline{\Sigma;\mathbb{M}, xS_By, yS_Aw;\Gamma} = \overline{\Sigma;\mathbb{M}, xS_By, yS_Aw, xS_Aw;\Gamma}$

4 $\mathrm{BL}^{\mathsf{G}}_{\mathsf{sf}}$: Soundness, Completeness and Termination

In Section, we present our theoretical resutls, showing that goal-directed search is sound and complete and deriving a depth bound on it.

Lemma 8 (Soundness of F-sequents) *Suppose the following hold.*

(a) $\Sigma;\mathbb{M};x : D \ll w : p \mid w_1 : g_1 \dots w_n : g_n$
(b) $\Sigma;\mathbb{M};\Gamma, x : D \Rightarrow w_i : g_i$*, for all* $1 \leq i \leq n$

Then, $\Sigma;\mathbb{M};\Gamma, x : D \Rightarrow w : p$

Proof. By induction on the height of the proof of hypothesis (a). See Appendix A, Lemma 17 for details.

Theorem 9 (Soundness) *The following hold*

A. If $\Sigma;\mathbb{M};\Gamma \Rightarrow x : \varphi$ *then* $\Sigma;\mathbb{M};\Gamma \Rightarrow x : \varphi$
B. If $\Sigma;\mathbb{M};\Gamma;\Xi \Leftarrow w : G$ *then* $\Sigma;\mathbb{M};\Gamma, \Xi \Rightarrow w : G$
C. If $\Sigma;\mathbb{M};\Gamma \Leftrightarrow w : p$ *then* $\Sigma;\mathbb{M};\Gamma \Rightarrow w : p$

Proof. By simultaneous induction on given derivations and case analysis of the last rule in them. See Appendix A, Lemma 18 for details.

Theorem 10 (Completeness) *The following hold.*

A. $\Sigma;\mathbb{M};\Gamma \Rightarrow x : G$ *implies* $\Sigma;\mathbb{M};\Gamma \Rightarrow x : G$
B. $\Sigma;\mathbb{M};\Gamma, \Xi \Rightarrow x : G$ *implies* $\Sigma;\mathbb{M};\Gamma;\Xi \Leftarrow x : G$

Proof. By simultaneous lexicographic induction, first on the depths of the given derivations, and then on the order (B) > (A). For (B) we also subinduct on $size(\Xi)$. More precisely, the following uses of the i.h. are legitimate:

– We are proving (B) and the i.h. is invoked for (A) or (B) with a derivation of smaller depth.
– We are proving (A) and the i.h. is invoked for (A) or (B) with a derivation of smaller depth.
– We are proving (B) and the i.h. is invoked for (A) with a derivation of equal depth.
– We are proving (B) and the i.h. is invoked for (B) with a derivation of equal depth and Ξ of smaller size.

See Appendix A, Theorem 31 for details.

Due that $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is s syntactic restriction of $\mathrm{BL}_{\mathrm{sf}}$, it comes with no surprise that $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is decidable. However, decidability of $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is not sufficient to establish that proof search in SeqG always terminated. The fact that SeqG is sound and complete w.r.t. SeqC does not necessarily mean that it is a decision procedure. However, by the countermodel producing decision procedure presented in [15], we know that there exists an upper bound k[2] to the depth of proof trees generated by backward application of SeqC rules.

Lemma 11 *If $\Sigma; \mathbb{M}; \Gamma \Rightarrow z : \varphi$ is provable with a derivation of at most depth k then $\Sigma; \mathbb{M}; \Gamma \Mapsto z : \varphi$ is provable with a derivation of at most depth $3k + 1$.*

Proof. The proof is by induction on the height of the derivation of $\Sigma; \mathbb{M}; \Gamma \Rightarrow$ and case analysis on the last applied rule. We present only two interesting cases
Case. $\dfrac{}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Rightarrow y : p}\,\mathrm{init}$
Here $k = 1$, to show that $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Mapsto y : p$ is provable with a depth at most of 4

1. $\overline{\Sigma; \mathbb{M}, x \leq y}; x : q \ll y : p \mid \cdot$, provable with depth 1 (Rule init)
2. $\overline{\Sigma; \mathbb{M}, x \leq y}; \Gamma, x : p \Leftrightarrow y : p$, provable with depth 2 (Rule (choice) on 1)
3. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Rightarrow y : p$, provable with depth 3 (Rule (atom) on 2)

Case. $\dfrac{\Sigma; \mathbb{M}; \Gamma \Rightarrow w : \alpha, w : \beta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow w : \alpha \vee \beta}\vee\mathrm{R}$
By inductive hypothesis and by Theorem 16 we have that either $\Sigma; \mathbb{M}; \Gamma \Mapsto w : \alpha$ or $\Sigma; \mathbb{M}; \Gamma \Mapsto w : \beta$ are provable with a derivation of at most height $3(k-1)+1$. Without loss of generality, suppose that $\Sigma; \mathbb{M}; \Gamma \Mapsto w : \alpha$. Hence, with an application of Rule $(\vee R_1)$ we obtain a proof of $\Sigma; \mathbb{M}; \Gamma \Mapsto w : \alpha \vee \beta$ with a derivation of at most height $3k + 1$.

Lemma 12 *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.*

[2] Notice that, in order to argue for the decidability of SeqG it is not necessary to exactly compute k.

Proof. The proof follows directly from [15] where it is presented a decision procedure for $\mathrm{BL}_{\mathrm{sf}}$.

Theorem 13 *Let φ a $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ formula and let k be the upperbound induced by the countermodel producing decision procedure in [15], then $\Sigma;\mathbb{M};\Gamma \Rightarrow z:\varphi$ is provable with a derivation of at most depth k if and only if $\Sigma;\mathbb{M};\Gamma \Mapsto z:\varphi$ is provable with a derivation of at most depth $3k+1$.*

Proof. The left to right direction follows from Theorem 9 while the opposite is given by Lemma 11.

Corollary 14 (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 12 for the starting sequent.*

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

In this Section we describe a sequent calculus based on SeqG that emits abducibles from unprovable sequents. The abduction procedure is presented as a calculus $\mathrm{SeqG}^{\mathrm{AB}}$ in Figure 4. The calculus is an extension of the calculus SeqG of Figure 3 with sequents of the form $\Sigma;\mathbb{M};\Gamma \Mapsto z:\varphi \searrow_m \Theta$ where Θ is a formula called *abducible* which represents the missing credentials (if any) that together with $\Sigma;\mathbb{M};\Gamma$ logically imply $z:\varphi$. Reading the rules backwards, the calculus is an algorithm with inputs $\Sigma;\mathbb{M};\Gamma$ and $z:\varphi$ and output Θ. To assure termination, we adopt an heuristic that bounds the depth of the proof trees generated by the $\mathrm{SeqG}^{\mathrm{AB}}$ to an integer m.

We observe that although $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ is decidable[3], the calculus SeqG does not always terminates. As a simple example consider the proof tree generated by SeqG with root sequent $x;x \leq x;x:p \to p \Mapsto x:p$, such tree has an infinite depth which is given by the rule (choice) rule which generates an infinite list of subgoals $x:p$. A main difference w.r.t. SeqG is the rule (AB) which, in case no other rule is applicable or the sequent has reached its maximum depth, outputs an abducible $\mathrm{AB}(\Sigma;\mathbb{M};\Gamma;\Delta)$, which is defined below. Here, $\mathrm{root}(\mathbb{M})$ is the root of the underlying tree of $\mathbb{M}$.

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

Intuitively, for a give goal $w:p$, we look at the path between the root of the underlying tree of $\mathbb{M}$ and y. Because the $\Sigma,\mathbb{M}$ and Γ are closed under backward application of rules (I), (mon-S) and (trans) (thanks to the rule atom), either $(\mathrm{root}(\mathbb{M})) \leq y \in \mathbb{M}$ or $(\mathrm{root}(\mathbb{M}))S_A y \in \mathbb{M}$ 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

[3] In fact, in [15] it is proved that $\mathrm{BL}_{\mathsf{sf}}$ has the finite model property and, hence, it is decidable.

suffices to add the credential A says p to complete the proof. If both sets in the definition of $\mathrm{AB}(\Sigma; \mathbb{M}; \Gamma; w : p)$ are empty, then $\mathrm{AB}(\Sigma; \mathbb{M}; \Gamma; w : p) = \bot$.

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:

- $F \models \top$ (always)
- $F \models p$ 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 $F \models \Theta_2$
- $F \models \Theta_1 \vee \Theta_2$ iff $F \models \Theta_1$ or $F \models \Theta_2$

The following theorem states that 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 15 (Soundness). *The following hold:*

A. Let $\Sigma; \mathbb{M}; \Gamma \Leftrightarrow w : p \searrow_m \Theta$ *and* $F \models \Theta$*, then* $\Sigma; \mathbb{M}; \Gamma \cup root(\mathbb{M}) : F \Mapsto w : p$
B. Let $\Sigma; \mathbb{M}; \Gamma; \Xi \Leftarrow w : G \searrow_m \Theta$ *and* $F \models \Theta$*, then* $\Sigma; \mathbb{M}; \Gamma, \Xi \cup root(\mathbb{M}) : F \Mapsto w : G$
C. Let $\Sigma; \mathbb{M}; \Gamma \Mapsto w : G \searrow_m \Theta$ *and* $F \models \Theta$*, then* $\Sigma; \mathbb{M}; \Gamma \cup root(\mathbb{M}) : F \Mapsto w : G$

Proof. By simultaneous lexicographic induction, first on the depths of the derivations and then on the order $A < B < C$. We show the two most interesting cases:

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

We are given that $F \models AB(\Sigma; \mathbb{M}; \Gamma; w : p)$, so one of the following must be true:

1. $(\mathrm{root}(\mathbb{M})) \leq w \in \mathbb{M}$ and $p \in F$: Then, it is sufficient to complete a proof of $\Sigma; \mathbb{M}; \Gamma, \mathrm{root}(\mathbb{M}) : F \Leftrightarrow w : p$ by applying the rule (init) on $\Sigma; \mathbb{M}; \mathrm{root}(\mathbb{M}) : p \ll w : p$.
2. $(\mathrm{root}(\mathbb{M}))S_A w \in \mathbb{M}$ and $(Ap \text{ says}) \in F$: Then, it is sufficient to complete a proof of $\Sigma; \mathbb{M}; \Gamma, root(\mathbb{M}) : F \Leftrightarrow w : p$ by:
   (a). $\Sigma; \mathbb{M}; \mathrm{root}(\mathbb{M}) : A \text{ says } p \ll w : p$ (rule choice on $\mathrm{root}(\mathbb{M}) : A$ says p)
   (b). $\Sigma; \mathbb{M}; w : p \ll w : p$ (rule says L)
   (c). $\Sigma; \mathbb{M}; w : p \ll w : p \mid \bullet$ (rule init)

5.1 A Prolog Implementation of $\mathrm{SeqG}^{\mathrm{AB}}$

The prover for $\mathrm{SeqG}^{\mathrm{AB}}$ has been implemented in Prolog and tested with two of its major interpreters: SWI and GNU. The main part of the program is a set of clauses, each one representing a sequent rule or axiom that, when queried with a formula expressed in $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ and together with Prolog's depth-first search mechanism, either proves the formula if it is indeed provable or gives a set of

abducibles that explain to the user what went wrong and what additional data to submit for a successful evaluation.

The top level predicates are either `prove(Formula)` if a boolean answer is all that is needed or `prove(Formula, Abducibles)` if a set of abducible is necessary. If the second predicate is chosen and the given formula is provable then the resultant set of abducibles will be empty and will be represented as `[]`.

The invocation of one of the `prove` instances starts the proving mechanism. In particular, the proof search starts with a predicate called `r_sequents` whose signature is `r_sequents(Formula,(Σ,𝕄,Γ,Γ_S,Ξ,Δ_S),Max_Depth,Used,Abducibles)` and whose parameters mean (respectively): the input formula, a tuple that forms the context for the formula, its elements are the set of labels (initially just the starting point "u"), the set of relations (initially just "$u \leq u$") and both sides of the context with the `_S` counterpart used to keep track of the history of applied rules. The Prolog proof search than proceeds by trying to recursively apply the clauses of the calculus in the following order: ($\top$R), (says R), ($\wedge$R), ($\vee$R), ($\rightarrow$ R) and (atom). The first one whose conditions are satisfied guides the subsequent steps of the proof.

As the implementation mirrors the rules in 4 the only clauses that do not recurse are those representing (atom) and ($\rightarrow$ R) whose execution leads to the predicates, respectively, `n_sequents` and `expand_l_sequents`.

The set of predicates under the name of `expand_l_sequents` and their auxiliary procedures `l_sequents` behave similarly to `r_sequents` with the search procedure trying in turn to satisfy the predicates that correspond to the following rules: ($\top$L), ($\bot$L), ($\wedge$L), ($\vee$L), (pr), (L2R).

As explained above, once the conditions for (*atom*) are satisfied the control goes to the predicate `n_sequents` but, right before this important call, the predicate `expand_sat_sequents` and its auxiliary procedures `sat_sequents` are used to saturate the contexts mimicking the rules: (mon-S), (relf), (trans), (I), (sf-refl), (sf), (sf-trans), (sf-unit), (sf-mon) in this precise order. Once this step is complete the `n_sequents` predicate sets up the stage for the one that implements the (*choice*) rule: `nd_choice`.

This predicate, along with the set formed by the `f_sequents` predicates representing (init), (sf), ($\wedge$ L), ($\rightarrow$ L), (says L), is the core of the abducibles extraction procedure.

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

In this Section we present an extension of the well-known `rsync` Unix program which includes a reference monitor able to reason about SeqG$^{\mathrm{AB}}$ policies. With Smart-rsync is possible to associate a $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ policy to resources and, therefore, to condition the synchronization of one or more files to the existence of a proof in SeqG$^{\mathrm{AB}}$ which implies that the corresponding sync request is compliant with the specified policy. In case the request of sync is not compliant with the policy associated to the resource, Smart-rsync is able to output the missing credentials

that are requested in order to grant access (i.e., the abducibles). The Smart-rsync architecture is based on the following main components:

- P-SeqG$^{\mathrm{AB}}$: Is the reference monitor (written in Prolog) that implements the calculus reported in Figure 4, details of the implementation together with an analysis of its efficiency are reported in Section 5.1.
- **rsyncd**: This is the standard `rsync` daemon which has to be active and properly configured on the machine that the user is trying to request a sync from.
- **srsyncd**: This is a server (written in C++) that receives principals' requests (together with a possibly empty set of credentials) and queries P-SeqG$^{\mathrm{AB}}$ to check whether the request can be granted or not.
- **srsync**: This is the client that performs the query against the server, the eventual process of requesting missing credentials and the final `rsync` call, in case permission is granted. The initial request to sync can result in two scenarios: the simplest one is when the credentials that the user already possess are enough to be granted the permission to sync, in this case the `srsync` call gets performed and the requested resources should be properly synced to the users' machine.
  On the other hand, if P-SeqG$^{\mathrm{AB}}$ is not able to close the proof tree it outputs a set of abducibles that are forwarded via srsyncd to the principal that issued the request. In this latter case, rsync tries to automatically obtain from other principals the necessary credentials to get the corresponding request authorized. In particular, it contacts all the corresponding principals appearing as indexes of says in the abducibles until a satisfactory response is obtained. If it is able to acquire these missing credentials then the original request is re-submitted together with the newly acquired credentials, otherwise it fails.
- **cred_serv**: This second kind of server is configured to handle only one policy at a time and is the one that responds to the queries in the abduction stage of the interaction. Every entity participating in the system can run an instance of this server in the background that will be queried by the other actors when they need to obtain some credentials from each other.

Finally, it is important to note that every message exchanged between principals via srsyncd and cred-serv is cryptographically signed in order to preserve authenticity.

6.1 Illustration of a Running Example

We now provide a detailed example to illustrate the workflow of the Smart-rsync architecture. The example is based on the scenario reported in Figure 5 in which there is a principal b that performs a request to a Smart-rsync server in order to synchronize with $file1$. The Smart-rsync server has a policy to check whether a principal is authorized to synchronize with $file1$ which is composed by two $\mathrm{BL}^{\mathsf{G}}_{\mathrm{sf}}$ formulae[4]:

[4] The propositional variable $sf1$ reads as "synchronize with $file1$".

1. a says $sf1 \rightarrow sf1$
2. a says $trusted\_b \rightarrow (a$ says $(b$ says $sf1 \rightarrow sf1))$

The first policy can be read as: "If principal a requests to synchronize with $file1$ then it is authorized"; while the second policy can be read as: "If principal a issues a credential which says that principal b is trusted then principal a says that b is authorized to synchronize with $file1$". The Smart-rsync server runs a standard rsync deamon of which we report the portion of its configuration file (i.e., `rsyncd.conf`) related to the sync of $file1$:

```
...
[sf1]
     path = /path/to/some/folder
     comment = Comment here
     read only = yes
     list = yes
     auth users = b, a
     secrets file = /path/to/rsyncd.secrets
...
```

where `sf1` is the alias for the resource pointed by the variable `path`. The file `rsyncd.secrets` contains the appropriate username, password association and, in particular, it contains the association between principal b and the password used to sync with $file1$:

```
...
b:password
...
```

Once that all the config files are ready we can finally launch the rsync daemon with the following command[5]:

```
# rsync --daemon --config=/path/to/rsyncd.conf
```

The following command, instead, launches the application `srsyncd`:

```
$ srsyncd --daemon --prikey keys/server_private.pem
          --pubkeys keys/ --policies policies/
          --port-number 3333 --log-level 2
```

which points the server to (respectively) its private key (previously generated by the `openssl genrsa` utility), the public keys of the principals authorized to interact with, the directory containing all the policies of the shared resources, the port number on which to listen in order to serve requests and the verbosity level of the logs. In particular, the policy regarding the sync with $file1$ is stored in `policies/sf1.pl` as follows:

```
policy(a controls sf1).
policy(a says trusted_b -> a trusts b on sf1).
```

[5] from a `root` shell

where "`a controls sf1`" is a shortcut for "`a says sf1 -> sf1`" and "`a trusts b on sf1`" stands for "`a says( b says sf1 -> sf1)`".

The remote principal a runs an instance of **cred_serv** which has been launched with the following command

```
$ cred_serv --daemon --prikey keys/a_private.pem
            --pubkeys keys/ --policy remote_examples/policy_a.pl
            --port-number 3334 --log-level 2
```

the options are very much like the ones described before for **srsyncd** and the relevant policy file, `policy_a.pl` contains:

```
policy(b says a says trusted_b -> a says trusted_b).
```

The above policy can be intuitively read as follows: "if principal b requests a credential form principal a supporting that b is trusted then principal a is authorized to send such credential". Concerning principal b, the request to synchronize with $file1$ is issued by `srsync` program by running the following command on the principal's machine:

```
$ srsync --whoami b --prikey keys/b_private.pem
         --pubkeys keys/
         --credentials remote_examples/credentials
         --request remote_examples/request
         --addresses remote_examples/addresses
```

which specifies the principal that is issuing the request (option `--whoami`), its associated private key (option `--prikey`), the credentials that are submitted together with the request (options `--credentials` and `--request`) and a list mapping names of principals with IP addresses and ports numbers. The execution of `srsync` will trigger the following sequence of events (reported in Figure 5):

1. Principal b sends, via `srsync`, to Smart-rsync server a request to synchronize with $file1$ by submitting `sf1` signed with its private key which encodes the formula b says $sf1$.
2. The `srsyncd` deamon forwards the request from principal b to P-SeqG$^{\text{AB}}$ to check whether from the policy associated with `sf1.pl` together with b says $sf1$ it can be proved that $sf1$ holds true.
3. The prover P-SeqG$^{\text{AB}}$ fails to prove $sf1$ and returns $sf1 \vee a$ says $trusted\_b$ as abducibles.
4. The `srsyncd` deamon sends to principal b the abducibles in the following form {`sf1;a says trusted_b`} where the `;` stands for logical disjunction $\vee$.
5. The `srsync` client associated to principal b selects from the set of abducibles those that are credentials (i.e., of the form c says $\cdot$) which, in this example, is {`a says trusted_b`}. Then, the client looks for the IP address associated with principal a in the file `remote_examples/addresses` and sends to principal a the request to receive the necessary credential `b says(a says trusted_b)`.
6. The server `cred_serv` of principal a queries P-SeqG$^{\text{AB}}$ to check whether from the policy specified in `policy_a.pl` and the request `b says(a says trusted_b)` it is possible to derive `a says trusted_b`. In this example, it is sufficient for b

to ask for the necessary credential in order to get it and, therefore, P-SeqG$^{\mathrm{AB}}$ succeeds in deriving `a says trusted_b`. Hence, `cred_serv` sends the message `trusted_b` signed with the private key of principal a.

7. Once that `srsync` receives the requested credential `a says trusted_b` it contacts again the Smart-rsync server by submitting the request to sync with file1 (i.e., `b says sf1`) together with the newly acquired credential. This time the prover P-SeqG$^{\mathrm{AB}}$ of the Smart-sync server succeeds in inferring `sf1` and, therefore, principal b can successfully execute the command `rsync -a b@RSYNCD_IP::sf1 sf1` to sync with $file1$.

7 Related Work

Although $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$ is a fragment of $\mathrm{BL}_{\mathrm{sf}}$, its goal-directed proof theory is intrinsically different from the one presented by GGR for $\mathrm{BL}_{\mathrm{sf}}$ [15]. Whereas the proof theory of $\mathrm{BL}_{\mathrm{sf}}$, SeqC, is a standard labeled sequent calculus, the proof theory of $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, SeqG, is a marriage of labeled sequent calculi with backchaining search, which we believe to be a novel contribution. SeqG works much faster in practice than SeqC, but some applications of SeqC presented by GGR, such as saturation and countermodel generation, are incompatible with goal-directed search and it seems difficult to adapt SeqG to cover those applications.

There has also been prior work on goal-directed search in a related logic BL, presented in Garg's thesis [10, 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 Garg's thesis (but can be expressed in $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$). Our proof of completeness of goal-directed search uses the same structure as that of Garg.

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 [20] 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 [7], based on Datalog with constraints, uses tabled backchaining search, which modifies backchaining search to decide Datalog in polynomial time. However, as discussed in Section 3, Datalog is much less expressive than $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$.

8 Conclusions

We have presented $\mathrm{BL}_{\mathrm{sf}}^{\mathsf{G}}$, a goal-directed fragment of $\mathrm{BL}_{\mathrm{sf}}$ [15], 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`.

R sequents

$$\frac{\overline{\Sigma; \mathbb{M}; \Gamma} \Leftrightarrow x : p}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : p}\text{atom} \qquad \frac{\Sigma, y; \mathbb{M}, xS_A y; \Gamma \Mapsto y : G}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : A \ \mathsf{says}\ G}\mathsf{says}\text{R} \qquad \frac{\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_i}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_1 \vee G_2}\vee\text{R}_i$$

$$\frac{}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : \top}\top\text{R} \qquad \frac{\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_1 \qquad \Sigma; \mathbb{M}; \Gamma \Mapsto x : G_2}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_1 \wedge G_2}\wedge\text{R}$$

$$\frac{\Sigma, y; \mathbb{M}, x \leq y; \Gamma; y : N \Leftarrow y : G}{\Sigma; \mathbb{M}; \Gamma \Mapsto x : N \to G}\to\text{R}$$

L sequents

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

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

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

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

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

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

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

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

R sequents

$$\frac{\overline{\Sigma; \mathbb{M}; \Gamma} \Leftrightarrow x : p \searrow_{m-1} \Theta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : p \searrow_m \Theta} atom \qquad \frac{\Sigma, y; \mathbb{M}, xS_A y; \Gamma \Rightarrow y : G \searrow_{m-1} \Theta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : A \text{ says } G \searrow_m \Theta} \text{says } R$$

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

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

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

L sequents

$$\frac{\Sigma; \mathbb{M}; \Gamma \Rightarrow w : G \searrow_{m-1} \Theta}{\Sigma; \mathbb{M}; \Gamma; \cdot \Leftarrow w : G \searrow_m \Theta} L2R \qquad \frac{\Sigma; \mathbb{M}; \Gamma, x : D; \Xi \Leftarrow w : G \searrow_{m-1} \Theta}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : D \Leftarrow w : G \searrow_m \Theta} pr$$

$$\frac{\Sigma; \mathbb{M}; \Gamma; \Xi \Leftarrow w : G \searrow_{m-1} \Theta}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : \top \Leftarrow w : G \searrow_m \Theta} \top \text{L} \qquad \frac{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1, x : N_2 \Leftarrow w : G \searrow_{m-1} \varphi}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1 \wedge N_2 \Leftarrow w : G \searrow_m \varphi} \wedge \text{L}$$

$$\frac{}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : \bot \Leftarrow w : G \searrow_m \top} \bot \text{L}$$

$$\frac{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1 \Leftarrow w : G \searrow_{m-1} \Theta_1 \qquad \Sigma; \mathbb{M}; \Gamma; \Xi, x : N_2 \Leftarrow w : G \searrow_{m-1} \Theta_2}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1 \vee N_2 \Leftarrow w : G \searrow_m \Theta_1 \wedge \Theta_2} \vee \text{L}$$

N sequents

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

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

F sequents

$$\frac{}{\Sigma; \mathbb{M}, x \leq w; x : q \ll w : p \mid \bullet} \text{init} \qquad \frac{}{\Sigma; \mathbb{M}; x : A \text{ sf } B \ll x : A \text{ sf } B \mid \bullet} \text{sf} \qquad \frac{\Sigma; \mathbb{M}; x : D_i \ll w : p \mid Gl}{\Sigma; \mathbb{M}; x : D_1 \wedge D_2 \ll w : p \mid Gl} \wedge \text{L}_i$$

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

Fig. 4. SeqG$^{\text{AB}}$: Abducibles extraction for BL$^{\mathsf{G}}_{\text{sf}}$

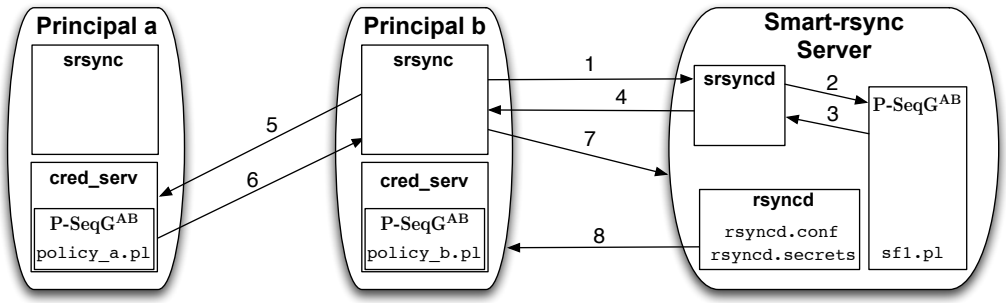

Fig. 5. The Smart-rsync Architecture

Bibliography

[1] This paper's technical report. Anonymously available at `http://dl.dropbox.com/u/18441484/tech_report_STM12_anonymized.pdf`

[2] Abadi, M.: Logic in access control. In: Procs. of LICS. pp. 228–233 (2003)

[3] Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A calculus for access control in distributed systems. ACM TPLS 15(4), 706–734 (1993)

[4] Avijit, K., Datta, A., Harper, R.: Distributed programming with distributed authorization. In: Procs. of TLDI. pp. 27–38 (2010)

[5] Bauer, L.: Access Control for the Web via Proof-Carrying Authorization. Ph.D. thesis, Princeton University (2003)

[6] Bauer, L., Garriss, S., McCune, J.M., Reiter, M.K., Rouse, J., Rutenbar, P.: Device-enabled authorization in the Grey system. In: Procs. of ISC. pp. 431–445 (2005)

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

[8] Becker, M.Y., Russo, A., Sultana, N.: Foundation of logic-based trust management. In: Procs. of IEEE Symposium on Security and Privacy (2012), to appear

[9] DeTreville, J.: Binder, a logic-based security language. In: Procs. of IEEE Symposium on Security and Privacy. pp. 105–113 (2002)

[10] Garg, D.: Proof Theory for Authorization Logic and Its Application to a Practical File System. Ph.D. thesis, Carnegie Mellon University (2009)

[11] Garg, D., Abadi, M.: A modal deconstruction of access control logics. In: Procs. of FoSSaCS. pp. 216–230 (2008)

[12] Garg, D., Pfenning, F.: Non-interference in constructive authorization logic. In: Procs. of CSF. pp. 283–293 (2006)

[13] Garg, D., Pfenning, F.: A proof-carrying file system. In: Procs. of IEEE Symposium on Security and Privacy. pp. 349–364 (2010)

[14] Genovese, V., Rispoli, D., Gabbay, D.M., van der Torre, L.: Modal access control logic: Axiomatization, semantics and FOL theorem proving. In: Procs. of STAIRS. IOS Press (2010)

[15] Genovese, V., Garg, D., Rispoli, D.: Labeled sequent calculi for access control logics: Countermodels, saturation and abduction. In: Procs. of CSF (2012), to appear. Available online at `http://www.mpi-sws.org/~dg/`

[16] Gurevich, Y., Neeman, I.: Logic of infons: The propositional case. ACM TOCL 12(2) (2011)

[17] Jia, L., Vaughan, J.A., Mazurak, K., Zhao, J., Zarko, L., Schorr, J., Zdancewic, S.: Aura: A programming language for authorization and audit. In: Procs. of ICFP. pp. 27–38 (2008)

[18] 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)

[19] Negri, S.: Proof analysis in modal logic. Journal of Philosophical Logic 34, 507–544 (2005)
[20] Pimlott, A., Kiselyov, O.: Soutei, a logic-based trust-management system. In: Procs. of FLOPS. pp. 130–145 (2006)
[21] Schneider, F.B., Walsh, K., Sirer, E.G.: Nexus Authorization Logic (NAL): Design rationale and applications. ACM TISSEC 14(1), 1–28 (2011)
[22] Swamy, N., Chen, J., Fournet, C., Strub, P.Y., Bhargavan, K., Yang, J.: Secure distributed programming with value-dependent types. In: Procs. of the 16th ACM SIGPLAN ICFP. pp. 266–278 (2011)
[23] 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.

A Proofs from Section

Theorem 16 (Height-preserving Disjunction Property) *If disjunction does not appear in negative positions and if* $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ *is derivable with a proof of height* h*, then there is some* $w : \varphi \in \Delta$ *such that* $\Sigma_w; \mathbb{M}_w; \Gamma_w \Rightarrow w : \varphi$ *is derivable with a proof of at most height* h*.*

Proof. By induction on the given derivation of $\Sigma; \mathbb{M}; \Gamma \Rightarrow \Delta$ and case analysis of its last rule. We show only the interesting cases.

Case 1. $\dfrac{}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Rightarrow y : p, \Delta}\text{init}$

Consider $y : p$, it is obvious that $\Sigma_y; (\mathbb{M}, x \leq y)_y; (\Gamma, x : p)_y \Rightarrow y : p$ is provable.

Case 2. $\dfrac{}{\Sigma; \mathbb{M}; \Gamma, x : A \mathsf{\ sf\ } B \Rightarrow x : A \mathsf{\ sf\ } B, \Delta}\text{sf}$

Consider $x : A \mathsf{\ sf\ } B$, it is obvious that $\Sigma_x; \mathbb{M}_x; (\Gamma, x : A \mathsf{\ sf\ } B)_x \Rightarrow x : A \mathsf{\ sf\ } B$ is provable.

Case 3. $\dfrac{}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \top, \Delta}\top\text{R}$

Clearly, $(\Sigma)_x; (\mathbb{M})_x; (\Gamma)_x \Rightarrow x : \top$ is provable.

Case 4. $\dfrac{}{\Sigma; \mathbb{M}; \Gamma, x : \bot \Rightarrow \Delta}\bot\text{L}$

Clearly, for any $y : \varphi \in \Delta$, $(\Sigma)_x; (\mathbb{M})_x; (\Gamma)_x, x_\bot \Rightarrow y : \varphi$ is provable.

Case 5. $\dfrac{\Sigma;\mathbb{M};\Gamma \Rightarrow x:\alpha, x:\alpha\wedge\beta,\Delta \qquad \Sigma;\mathbb{M};\Gamma \Rightarrow x:\beta, x:\alpha\wedge\beta,\Delta}{\Sigma;\mathbb{M};\Gamma \Rightarrow x:\alpha\wedge\beta,\Delta}\wedge\,\mathrm{R}$

By induction hypothesis, if there exists a $w:\varphi\in\Delta$, with $w\neq x$ such that $\Sigma_w;\mathbb{M}_w;\Gamma_w \Rightarrow w:\varphi$ we are done. On the other hand if we can prove both $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\alpha$ and $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\beta$ then we obtain, by applying forward $(\wedge R)$, $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\alpha\wedge\beta$.

Case 6. $\dfrac{\Sigma;\mathbb{M};\Gamma \Rightarrow x:\alpha, x:\beta, x:\alpha\vee\beta,\Delta}{\Sigma;\mathbb{M};\Gamma \Rightarrow x:\alpha\vee\beta,\Delta}\vee\,\mathrm{R}$

By induction hypothesis, we have that there exists a $w:\varphi\in(x:\alpha, x:\beta,\Delta)$ such that $\Sigma_w;\mathbb{M}_w;\Gamma_w \Rightarrow w:\varphi\in\Delta$. If $w:\varphi\in\Delta$ then we are done. If $\Sigma_x;\mathbb{M}_x;\Gamma_w \Rightarrow x:\alpha$ then, by weakening, we have $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\alpha, x:\beta$ and, by an application of $\vee R$, we finally get $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\alpha\vee\beta$.

Case 7. $\dfrac{\Sigma, y;\mathbb{M}, x\leq y;\Gamma, y:\alpha \Rightarrow y:\beta,\Delta}{\Sigma;\mathbb{M};\Gamma \Rightarrow x:\alpha\to\beta,\Delta}\to\,\mathrm{R}$

By induction hypothesis we have that there exists a $w:\varphi\in(y:\beta,\Delta)$ such that $(\Sigma, y)_w;(\mathbb{M}, x\leq y)_w;(\Gamma, y:\alpha)_w \Rightarrow w:\varphi$, we then have two sub-cases:

1. $w:\varphi\in\Delta$ then, due that y is fresh, we have that $(\Sigma, y)_w=\Sigma_w$, $(\mathbb{M}, x\leq y)_w=\mathbb{M}_w$ and $(\Gamma, y:\alpha)_w=\Gamma_w$. Hence,

   $$\Sigma_w;\mathbb{M}_w;\Gamma_w \Rightarrow w:\alpha$$

   is provable.
2. $w:\varphi = y:\beta$ then, de that y is fresh, we have that $(\Sigma, y)_y=(\Sigma_y, y)$, $(\mathbb{M}, x\leq y)_y=(\mathbb{M}_y, x\leq y)$ and $(\Gamma,\ y:\alpha)_y=(\Gamma_y, y:\alpha)$. Hence we have that the following sequent is provable

   $$\Sigma_y, y;\mathbb{M}_y, x\leq y;\Gamma_y, y:\alpha \Rightarrow y:\beta$$

   Now, by applying forward $\to R$ and because y is fresh, we have $\Sigma_x;\mathbb{M}_x;\Gamma_x \Rightarrow x:\alpha\to\beta$

Case 8. $\dfrac{(3)\Sigma;\mathbb{M}, x\leq y;\Gamma, x:\alpha\to\beta \Rightarrow y:\alpha,\Delta \quad (2)\Sigma;\mathbb{M}, x\leq y;\Gamma, x:\alpha\to\beta, y:\beta \Rightarrow \Delta}{\Sigma;\mathbb{M}, x\leq y;\Gamma, x:\alpha\to\beta \Rightarrow \Delta}$

By induction hypothesis on (2) there must be a labelled formula $w:\varphi\in\Delta$ such that $\Sigma_w;(\mathbb{M}, x\leq y)_w;(\Delta, x:\alpha\to\beta, y:\beta)_w \Rightarrow w:\varphi$ we have several cases

1. If w is not ancestor of y (i.e., $w\notin\Sigma_y$) then we have $y:\beta\notin(\Gamma, x:\alpha\to\beta, y:\beta)_w$ and then we have that the following sequent is provable

   $$\Sigma_w;(\mathbb{M}, x\leq y)_w;(\Gamma, x:\alpha\to\beta)_w \Rightarrow w:\alpha$$

2. If w is an ancestor of y then we can have two sub-cases

– w is an ancestor of some $z \neq y$ such that $z \leq y$ (this includes that case in which $z = x$) then we have $y : \beta \notin (\Gamma, x : \alpha \to \beta, y : \beta)_w$ and then we conclude as in point 1.
– $w = y$ in this case, by induction hypothesis, we have that the following sequent is provable

$$(4)\Sigma_y; \mathbb{M}_y, x \leq y; \Gamma_y, x : \alpha \to \beta, y : \beta \Rightarrow y : \varphi$$

we then have to apply the inductive hypothesis on sequent (2)
- If there exists some $w : \varphi_1 \in \Delta$ such that

$$\Sigma_w; (\mathbb{M}, x \leq y)_w; (\Gamma, x : \alpha \to \beta)_w \Rightarrow w : \varphi_1$$

then we have proved the thesis.
- If the following sequent is provable

$$\Sigma_y; \mathbb{M}_y, x \leq y; \Gamma_y, x : \alpha \to \beta \Rightarrow y : \alpha$$

then, by weakening, also the following sequent is provable

$$(5)\Sigma_y; \mathbb{M}_y, x \leq y; \Gamma_y, x : \alpha \to \beta \Rightarrow y : \alpha, y : \varphi$$

Now, by applying forward $\to L$ on (4) and (5) we get

$$\Sigma_y; \mathbb{M}_y, x \leq y; \Gamma_y, x : \alpha \to \beta \Rightarrow y : \varphi$$

Case 9. $\dfrac{\Sigma, y; \mathbb{M}, xS_Ay; \Gamma \Rightarrow y : \alpha, \Delta}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : A \text{ says } \alpha, \Delta}\text{says R}$

Now by induction hypothesis we have that there exists a $w : \varphi \in (y : \alpha, \Delta)$ such that the following sequent is provable

$$(\Sigma, y)_w; (\mathbb{M}, xS_Ay)_w; \Gamma_w \Rightarrow w : \varphi$$

We distinguish two cases

1. $w : \varphi \in \Delta$, this case is trivial due that y is a fresh variable hence we have that sequent $\Sigma_w; \mathbb{M}_w; \Gamma_w \Rightarrow w : \varphi$ is provable.
2. In case $w : \varphi = y : \alpha$ we have that $(\Sigma_y, y) = \Sigma_x, y$, $(\mathbb{M}, xS_Ay)_y = \mathbb{M}_x, xS_Ay$ and $(\Gamma, y : \alpha)_y = \Gamma_x, y : \alpha$ which implies that the following sequent is derivable

$$\Sigma_x, y; \mathbb{M}_x, xS_Ay; \Gamma_x, y : \alpha$$

Now, with an application (looking forward) of says R we infer

$$\Sigma_x; \mathbb{M}_x; \Gamma_x \Rightarrow x : A \text{ says } \alpha$$

Lemma 17 (Soundness of F-sequents) *Suppose the following hold.*

(a) $\Sigma; \mathbb{M}; x : D \ll w : p \mid w_1 : g_1 \ldots w_n : g_n$
(b) $\Sigma; \mathbb{M}; \Gamma, x : D \Rightarrow w_i : g_i$, *for all* $1 \leq i \leq n$

Then, $\Sigma; \mathbb{M}; \Gamma, x : D \Rightarrow w : p$

Proof. By induction on the height of the proof of hypothesis (a). We case analyze the last rule in the derivation.

Case. $\dfrac{}{\Sigma; \mathbb{M}, x \leq w; x : q \ll w : p \mid \cdot}\text{init}$

Trivially, from rule (init) $\Sigma; \mathbb{M}, x \leq w; \Gamma, x : p \Rightarrow w : p$ is provable.

Case. $\dfrac{}{\Sigma; \mathbb{M}; x : A \ \mathsf{sf}\ B \ll x : A \ \mathsf{sf}\ B \mid \bullet}\text{sf}$

Trivially, from rule (sf) $\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf}\ B \Rightarrow x : A\mathsf{B}$ is provable.

Case. $\dfrac{\Sigma; \mathbb{M}; x : D_1 \ll w : p \mid w_1 : g_1 \ldots w_n : g_n}{\Sigma; \mathbb{M}; x : D_1 \wedge D_2 \ll w : p \mid w_1 : g_1 \ldots w_n : g_n} \wedge \text{L}_1$

To show $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Rightarrow w : p$

1. $\Sigma; \mathbb{M}; \Gamma, x : D_1 \Rightarrow w : p$ (inductive hypothesis)
2. $\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2 \Rightarrow w : p$ (By Weakening on 1)
3. $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Rightarrow w : p$ (Rule ($\wedge L$) on 2)

Case. $\dfrac{\Sigma; \mathbb{M}; x : D_2 \ll w : p \mid w_1 : g_1 \ldots w_n : g_n}{\Sigma; \mathbb{M}; x : D_1 \wedge D_2 \ll w : p \mid w_1 : g_1 \ldots w_n : g_n} \wedge \text{L}_2$

Similar to the previous case.

Case. $\dfrac{\Sigma; \mathbb{M}, x \leq y; y : D \ll w : p \mid w_1 : g_1 \ldots w_n : g_n}{\Sigma; \mathbb{M}, x \leq y; x : G \to D \ll w : p \mid y : G, w_1 : g_1 \ldots w_n : g_n} \to \text{L}$

To show $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow w : p$

1. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow y : G$ (hypothesis (b))
2. $(\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow w_i : g_i)_{i \in \{1 \ldots n\}}$ (hypothesis (b))
3. $\Sigma; \mathbb{M}, x \leq y; \Gamma, y : D \Rightarrow_D w : p$ (inductive hypothesis)
4. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D, y : D \Rightarrow w : p$ (be Weakening on 3)
5. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow w : p$ (by applying ($\to L$) on 4 and 1)

Case. $\dfrac{\Sigma; \mathbb{M}, xS_Ay; y : D \ll w : p \mid w_1 : g_1 \ldots w_n : g_n}{\Sigma; \mathbb{M}, xS_Ay; x : A \ \mathsf{says}\ D \ll w : p \mid w_1 : g_1 \ldots w_n : g_n} \mathsf{says}\ \text{L}$

To show $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \ \mathsf{says}\ D \Rightarrow w : p$

1. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, y : D \Rightarrow w : p$ (inductive hypothesis on (b))
2. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \ \mathsf{says}\ D, y : D \Rightarrow w : p$ (by Weakening on 1)
3. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \ \mathsf{says}\ D \Rightarrow w : p$ (by applying ($\mathsf{says}\ L$) on 2)

Theorem 18 (Soundness) *The following hold*

A. If $\Sigma; \mathbb{M}; \Gamma \Rightarrowtriangle x : \varphi$ *then* $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : \varphi$
B. If $\Sigma; \mathbb{M}; \Gamma; \Xi \Leftarrow w : G$ *then* $\Sigma; \mathbb{M}; \Gamma, \Xi \Rightarrow w : G$
C. If $\Sigma; \mathbb{M}; \Gamma \Leftrightarrow w : p$ *then* $\Sigma; \mathbb{M}; \Gamma \Rightarrow w : p$

Proof. By simultaneous induction on given derivations and case analysis of the last rule in them. We illustrate below some representative cases.

Case. $\dfrac{\Sigma, y; \mathbb{M}, xS_Ay; \Gamma \Rightarrow y : G}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : AG \text{ says}}\text{says R}$

To show $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : A \text{ says } G$

1. $\Sigma; \mathbb{M}, xS_Ay; \Gamma \Rightarrow y : G$ (inductive hypothesis on premise)
2. $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : A \text{ says } G$ (Rule (saysR) on 1)

Case. $\dfrac{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1 \quad \Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_2}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1 \wedge G_2} \wedge R$

To show $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1 \wedge G_2$

1. $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1$ (inductive hypothesis on premise)
2. $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_2$ (inductive hypothesis on premise)
3. $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1 \wedge G_2$ (Rule ($\wedge$ L) on 1 and 2)

Case. $\dfrac{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1, x : N_2 \Leftarrow w : G}{\Sigma; \mathbb{M}; \Gamma; \Xi, x : N_1 \wedge N_2 \Leftarrow w : G} \wedge L$

To show $\Sigma; \mathbb{M}; \Gamma, \Xi, x : N_1 \wedge N_2 \Rightarrow w : G$

1. $\Sigma; \mathbb{M}; \Gamma, \Xi, x : N_1, x : N_2 \Rightarrow w : G$ (by inductive hypothesis on premise)
2. $\Sigma; \mathbb{M}; \Gamma, \Xi, x : N_1 \wedge N_2 \Rightarrow w : G$ (by Rule ($\wedge$ L))

Case. $\dfrac{x : D \in \Gamma \quad \Sigma; \mathbb{M}; x : D \ll w : p \mid w_1 : g_1 \ldots w_n : g_n \quad (\Sigma; \mathbb{M}; \Gamma; \cdot \Rightarrow w_i : g_i)_{i=1}^n}{\Sigma; \mathbb{M}; \Gamma \Leftrightarrow w : p}\text{choice}$

To show $\Sigma; \mathbb{M}; \Gamma \Rightarrow w : p$

1. $(\Sigma; \mathbb{M}; \Gamma \Rightarrow w_i : g_i)_{i=1}^n$ (by i.h. on 3rd premise)
2. $\Sigma; \mathbb{M}; \Gamma, x : D \Rightarrow w : p$ (Lemma 8 on 2nd premise and point 1)
3. $\Sigma; \mathbb{M}; \Gamma \Rightarrow w : p$ (Contraction Theorem on 2. using 1st premise)

Lemma 19 (Admissibility of ($\rightarrow$ L)) *$\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D \Rightarrow y : G$ provable with a derivation of height h and*

> *A. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D, y : D \Rightarrow z : G'$ provable with a derivation of height k implies $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D \Rightarrow z : G'$ provable with a derivation of height at most $max(h, k)$*
> *B. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D, y : D; \Xi \Leftarrow z : G'$ provable with a derivation of height k implies $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D; \Xi \Leftarrow z : G'$ provable with a derivation of height at most $max(h, k)$*
> *C. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D, y : D \Leftrightarrow z : p$ provable with a derivation of height k implies $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \rightarrow D \Leftrightarrow z : p$ provable with a derivation of height at most $max(h, k)$.*

Proof. By simultaneous induction on the depths of the derivations given in (A)-(C) and case analysis of the last rule in them.

Lemma 20 (Admissibility of says L) *A.* $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D, y : D \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D \Rightarrow z : G$ *provable with a derivation of height at most* h.

B. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D, y : D; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D; \Xi \Leftarrow z : G$ *provable with a derivation of height at most* h.

C. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D, y : D \Leftrightarrow z : p$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \text{ says } D \Leftrightarrow z : p$ *provable with a derivation of height at most* h.

Proof. By simultaneous induction on the depths of derivation given in (A)-(C) and case analysis of their last rules.

Lemma 21 (Admissibility of $\wedge L$) *A.* $\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2 \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Rightarrow z : G$ *provable with a derivation of at most height* h.

B. $\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies implies* $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h.

C. $\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2 \Leftrightarrow z : p$ *provable with a derivation of height* h *implies implies* $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Leftrightarrow z : p$ *provable with a derivation of at most height* h.

Proof. By simultaneous induction on the depths of derivation given in (A)-(C) and case analysis of their last rules.

Lemma 22 (Admissibility of mon-S) *A.* $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w, xS_Aw; \Gamma \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w; \Gamma \Rightarrow z : G$ *provable with a derivation of at most height* h.

B. $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w, xS_Aw; \Gamma; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w; \Gamma; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h.

Proof. By simultaneous induction on the depths of derivations given in (A)-(B) and case analysis of their last rules. There is only one interesting case, that is shown below.

Case 10.
$$\frac{\overline{(\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w, xS_Aw); \Gamma \Leftrightarrow u : p}}{\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w; \Gamma \Rightarrow u : p}\text{atom}$$

To show $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w; \Gamma \Rightarrow u : p$

1. $\overline{(\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w, xS_Aw; \Gamma)} = \overline{(\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w)}$ (Defn.)
2. $\overline{(\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w)}; \Gamma \Leftrightarrow u : p$, with height $h-1$ (1st premise and 1)
3. $\Sigma; \mathbb{M}, x \leq y, yS_Az, z \leq w; \Gamma \Rightarrow u : p$, with height h (Rule (atom) on 2)

Lemma 23 (Admissibility of refl) *A.* $\Sigma, x; \mathbb{M}, x \leq x; \Gamma \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma, x; \mathbb{M}; \Gamma \Rightarrow z : G$ *provable with a derivation of at most height* h.

B. $\Sigma, x; \mathbb{M}, x \leq x; \Gamma; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma, x; \mathbb{M}; \Gamma; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h.

Lemma 24 (Admissibility of trans) *A.* $\Sigma; \mathbb{M}, x \leq y, y \leq z, x \leq z; \Gamma \Rightarrow w : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, x \leq y, y \leq z; \Gamma \Rightarrow w : G$ *provable with a derivation of at most height* h.

B. $\Sigma; \mathbb{M}, x \leq y, y \leq z, x \leq z; \Gamma; \Xi \Leftarrow w : G$ *provable with a derivation of height* h. *implies* $\Sigma; \mathbb{M}, x \leq y, y \leq z; \Gamma; \Xi \Leftarrow w : G$ *provable with a derivation of at most height* h.

Lemma 25 (Admissibility of I) *A.* $\Sigma; \mathbb{M}, xS_By, yS_Az, xS_Az; \Gamma \Rightarrow w : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_By, yS_Az; \Gamma \Rightarrow w : G$ *provable with a derivation of at most height* h.

B. $\Sigma; \mathbb{M}, xS_By, yS_Az, xS_Az; \Gamma; \Xi \Leftarrow w : G$ *provable with a derivation of height* h *implies*
$\Sigma; \mathbb{M}, xS_By, yS_Az; \Gamma; \Xi \Leftarrow w : G$ *provable with a derivation of at most height* h.

Lemma 26 (Admissibility of refl-sf) *A.* $\Sigma, x; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ A \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma, x; \mathbb{M}; \Gamma \Rightarrow z : G$ *provable with a derivation of at most height* h.

B. $\Sigma, x; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ A; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma, x; \mathbb{M}; \Gamma; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h.

Proof. By simultaneous induction on the depths of derivations given in (A)-(B) and case analysis of their last rules. There is only one interesting case, that is shown below.

Case 11. $\dfrac{\overline{(\Sigma, x; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ A)} \Leftrightarrow u : p}{\Sigma, x; \mathbb{M}; \Gamma \Rightarrow u : p}$refl-sf

To show $\Sigma, x; \mathbb{M}; \Gamma \Rightarrow u : p$

1. $\overline{(\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ A)} = \overline{(\Sigma, x; \mathbb{M}; \Gamma)}$ (Defn.)
2. $\overline{(\Sigma, x; \mathbb{M}; \Gamma)} \Leftrightarrow u : p$, with height $h - 1$ (1st premise and 1)
3. $\Sigma, x; \mathbb{M}; \Gamma \Rightarrow u : p$, with height h (Rule (refl-sf) on 2)

Lemma 27 (Admissibility of basic-sf) *A.* $\Sigma; \mathbb{M}, xS_By, xS_Ay; \Gamma, x : A \ \mathsf{sf} \ B \Rightarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_By; \Gamma, x : A \ \mathsf{sf} \ B \Rightarrow z : G$ *provable with a derivation of height at most* h.

B. $\Sigma; \mathbb{M}, xS_By, xS_Ay; \Gamma, x : A \ \mathsf{sf} \ B; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_By; \Gamma, x : A \ \mathsf{sf} \ B; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h.

Lemma 28 (Admissibilty of trans-sf) *A.* $\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ B, x : B \ \mathsf{sf} \ C, x : A \ \mathsf{sf} \ C \Rightarrow z : G$ *provable with derivation of height* h *implies* $\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf} \ B, x : B \ \mathsf{sf} \ C \Rightarrow z : G$ *provable with a derivation of at most height* h.

B. $\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf}\ B, x : B \ \mathsf{sf}\ C, x : A \ \mathsf{sf}\ C; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}; \Gamma, x : A \ \mathsf{sf}\ B, x : B \ \mathsf{sf}\ C; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h*.*

Lemma 29 (Admissibilty of mon1-sf) *A.* $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : A \ \mathsf{sf}\ B, y : A \ \mathsf{sf}\ B \Mapsto z : G$ *provable with derivation of height* h *implies* $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : A \ \mathsf{sf}\ B \Mapsto z : G$ *provable with a derivation of at most height* h*.*

B. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : A \ \mathsf{sf}\ B, y : A \ \mathsf{sf}\ B; \Xi \Leftarrow z : G$ *provable with derivation of height* h *implies* $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : A \ \mathsf{sf}\ B; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h*.*

Lemma 30 (Admissibilty of mon2-sf) *A.* $\Sigma; \mathbb{M}, xS_C y; \Gamma, x : A \ \mathsf{sf}\ B, y : A \ \mathsf{sf}\ B \Mapsto z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_C y; \Gamma, x : A \ \mathsf{sf}\ B \Mapsto z : G$ *provable with a derivation of at most height* h*.*

B. $\Sigma; \mathbb{M}, xS_C y; \Gamma, x : A \ \mathsf{sf}\ B, y : A \ \mathsf{sf}\ B; \Xi \Leftarrow z : G$ *provable with a derivation of height* h *implies* $\Sigma; \mathbb{M}, xS_C y; \Gamma, x : A \ \mathsf{sf}\ B; \Xi \Leftarrow z : G$ *provable with a derivation of at most height* h*.*

Theorem 31 (Completeness) *The following hold.*

A. $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G$ *implies* $\Sigma; \mathbb{M}; \Gamma \Mapsto x : G$
B. $\Sigma; \mathbb{M}; \Gamma, \Xi \Rightarrow x : G$ *implies* $\Sigma; \mathbb{M}; \Gamma; \Xi \Leftarrow x : G$

Proof. By simultaneous lexicographic induction, first on the depths of the given derivations, and then on the order (B) > (A). For (B) we also subinduct on $size(\Xi)$. More precisely, the following uses of the i.h. are legitimate:

- We are proving (B) and the i.h. is invoked for (A) or (B) with a derivation of smaller depth.
- We are proving (A) and the i.h. is invoked for (A) or (B) with a derivation of smaller depth.
- We are proving (B) and the i.h. is invoked for (A) with a derivation of equal depth.
- We are proving (B) and the i.h. is invoked for (B) with a derivation of equal depth and Ξ of smaller size.

To prove(A), we case analyze the last rule in the given derivation of $\Sigma; \mathbb{M}; \Gamma \Rightarrow_D x : \varphi$. For right rules we apply the i.h. to premises and then apply the corresponding rule from R-sequents. For left rules, semantical rules and access control rules, we apply the i.h. to the premises, and use one of Lemmas 19–30. Some representative cases are shown below.

Case. $\dfrac{}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Rightarrow y : p}\,\text{init}$
To show $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Mapsto y : p$.

1. $\overline{\Sigma; \mathbb{M}, x \leq y}; x : q \ll y : p \mid \cdot$ (Rule init)
2. $\overline{\Sigma; \mathbb{M}, x \leq y}; \Gamma, x : p \Leftrightarrow y : p$ (Rule (choice) on 1)

3. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : p \Mapsto y : p$ (Rule (atom) on 2)

Case. $\dfrac{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow y : G, z : G' \;\; \Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D, y : D \Rightarrow z : G'}{\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow z : G'} \to \mathrm{L}$

To show $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Mapsto z : G'$, by Theorem 16 we have to consider two subcases

Subcase $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow z : G'$

1. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Mapsto z : G'$ (inductive hypothesis on subcase)

Subcase $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Rightarrow y \; G$

1. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Mapsto y : G$ (i.h. (A) on 1st premise)
2. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D, y : D \Mapsto z : G'$ (i.h. (A) on 2nd premise)
3. $\Sigma; \mathbb{M}, x \leq y; \Gamma, x : G \to D \Mapsto z : G'$ (Lemma 19 on 1 and 2)

Case. $\dfrac{\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \;\mathsf{says}\; D, y : D \Rightarrow z : G}{\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \;\mathsf{says}\; D \Rightarrow z : G} \mathsf{says}\ \mathrm{L}$

To show: $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \;\mathsf{says}\; D \Mapsto z : G$.

1. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \;\mathsf{says}\; D, y : D \Mapsto z : G$ (i.h. (A) on premise)
2. $\Sigma; \mathbb{M}, xS_Ay; \Gamma, x : A \;\mathsf{says}\; D \Mapsto z : G$ (Lemma 20 on 1)

Case. $\dfrac{\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2 \Rightarrow z : G}{\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Rightarrow z : G} \wedge\mathrm{L}$

To show $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Mapsto z : G$

1. $\Sigma; \mathbb{M}; \Gamma, x : D_1, x : D_2 \Mapsto z : G$ (i.h. (A) on premise)
2. $\Sigma; \mathbb{M}; \Gamma, x : D_1 \wedge D_2 \Mapsto z : G$ (Lemma 21 on 1)

Case. $\dfrac{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1, x : G_2}{\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1 \vee G_2} \vee\mathrm{L}$

By Theorem 16 we have to consider two subcases

Subcase $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_1$

1. $\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_1$ (i.h. on subcase)
2. $\Sigma; \mathbb{M}; \Gamma \Mapsto x : G_1 \vee G_2$ (Rule $\vee R_1$ on 1)

The subcase $\Sigma; \mathbb{M}; \Gamma \Rightarrow x : G_2$ is similar and left to the reader