Stateful Authorization Logic
– Proof Theory and a Case Study

Deepak GARG [a,1] and Frank PFENNING [b]

[a] *Max Planck Institute for Software Systems, Germany*

[b] *Carnegie Mellon University, USA*

Abstract. We present the design, proof theory and metatheory of a logic for representing and reasoning about authorization policies. A salient feature of the logic, BL, is its support for system state in the form of interpreted predicates, upon which authorization policies often rely. In addition, BL includes Abadi et al.'s "says" connective and explicit time. BL is illustrated through a case study of policies for sharing sensitive information created in the U.S. intelligence community. We discuss design choices in the interaction between state and other features of BL and validate BL's proof theory by proving standard metatheoretic properties like admissibility of cut.

Keywords. Authorization logic, proof theory, stateful policies

[1]Corresponding author. Address: Max Planck Institute for Software Systems, Campus E1 5, D-66123 Saarbrücken, Germany. Email: dg@mpi-sws.org.

1. Introduction

Many authorization policies rely on conditions that are controlled by the environment and whose changes are not stipulated by the policies themselves. For example, a sensitive file may be accessible to the public **if the file is marked unclassified**; an employee may enter her office **if it is between 9 AM and 5 PM on a weekday**; a doctor may read any patient's health records **if there is a medical emergency**. Conditions such as "if the file is marked unclassified", written in boldface in the previous sentence, have the following characteristics: (a) They influence consequences of the authorization policy of interest, and (b) The authorization policy itself does not stipulate when or how such conditions change, although, in conjunction with other enforcement mechanisms in the system, it may constrain who may change them (e.g., the list of individuals who may mark a file unclassified may be stipulated by the authorization policy itself). We informally call conditions satisfying these two criteria *stateful*, and any authorization policy relying on them a stateful authorization policy.

Many formal proposals for representing, enforcing and reasoning about authorization policies use logic to represent the policies. Central to such use of logic is *proof theory*, which is used both to enforce the authorization policies through proof-carrying authorization or PCA [3, 5, 6], and to facilitate logical inference to analyze their consequences [11, 12]. Yet, despite several papers on proof theory of authorization logics without state [1, 14, 18], to the best of our knowledge, there has been no systematic work on proof theory for logics that can represent stateful authorization policies. The main objective of this paper is to fill this gap: We examine in detail the proof theory of a new logic BL in which stateful authorization policies can be represented.[2] We validate the logic's foundations by proving several metatheoretic properties of its proof system including admissibility of cut, which is a proof-theoretic statement of the logic's soundness [26, 32]. Empirically, we illustrate BL and justify its expressiveness through a case study of policies for sharing of sensitive U.S. intelligence information. Further, we discuss subtle design choices in the interaction between state and other components of the logic. Orthogonal to our main objective, we provide a new interpretation for the common access control connective k says s [2], which makes it easier to express a common form of delegation in the logic (Section 5.4).

At its core, BL is a first-order intuitionistic logic. To that we add the connective k says s, which means that principal k supports statement s, and *state predicates*, a subclass of predicates that can be verified through procedures external to the logic that may refer to the system state. Finally, in order to represent real time, upon which policies often rely, we include the connective $s @ [u_1, u_2]$ from our prior work with DeYoung [14]. $s @ [u_1, u_2]$ means that formula s holds in the time interval $[u_1, u_2]$, but possibly not outside of it. Through its combination of state predicates, explicit time (the @ connective), and the says connective, BL is a very expressive authorization logic.

There are two main challenges in incorporating state in an authorization logic like BL. The first is to decide the interaction between state predicates and other features of the logic, especially explicit time. For example, we may choose to treat time as a special case of the logic's general treatment of state and thus eliminate the need for the connective @. Some prior work follows this approach by providing in the logic a

---

[2]BL is an abbreviation for <u>B</u>inder <u>L</u>ogic. This name is a tribute to the authorization policy language Binder [13], whose design inspired the early stages of our work.
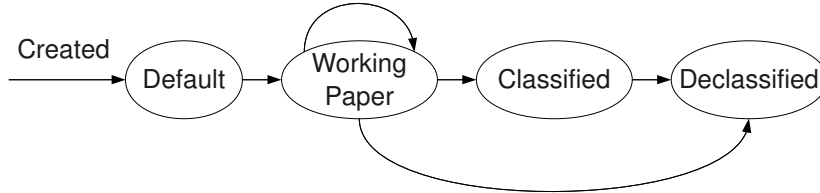
Figure 1. Stages of a sensitive intelligence file in the U.S.

constant, say localtime, that evaluates to the time at which the policy is interpreted [5, 8]. However, as we argue in Section 5.3, this results in a loss of expressiveness as compared to providing the connective @ in the logic explicitly, because any policy that refers to more than one point of time cannot be expressed if time is considered a part of the state. As a result, we include both the @ connective and stateful atoms in BL. Another key decision is the interpretation of the formula $s @ [u_1, u_2]$ when s is a stateful atom. The obvious interpretation is that the stateful atom s hold throughout the time interval $[u_1, u_2]$. However, as we illustrate and explain in Section 3.3.1, this interpretation allows representation of authorization policies that cannot be enforced by any reference monitor and, to prevent representation of such policies, we make a deliberate decision to use a non-obvious interpretation of $s @ [u_1, u_2]$ when s is a stateful atom.

The second challenge in incorporating state is the integration of external procedures for checking state predicates with the inference rules of the logic without breaking metatheoretic properties like admissibility of cut. In this regard, our proof theory is guided by, and similar to, prior work on integrating decision procedures for constraint domains into a logic [22, 33]. The key idea is to formally represent the external procedure for checking state predicates by an abstract judgment $E \models i$ (any environment which satisfies all stateful atoms in E, also satisfies the stateful atom i), and stipulating only a few, reasonable conditions on the judgment that are enough to obtain metatheoretic properties of interest, including admissibility of cut. We list these conditions in Section 3.2.

The rest of this paper is organized as follows. In Section 2, we present a part of our case study to motivate the need for state in authorization policies and also to illustrate, by way of example, the syntax and features of BL. In Section 3, we introduce the logic BL and its proof theory incrementally. We start with a first-order intuitionistic logic with the connective k says s, then add state predicates, and finally add explicit time. Section 4 presents metatheoretic properties of the proof system. Additional design choices, including the separation of state from time and our new interpretation of says, are discussed in Section 5. Section 6 revisits the case study with more details. Related work is discussed in Section 7 and Section 8 concludes the paper. Full proofs of theorems are presented in the first author's thesis [17] and the entire case study appears in a concise form in a separate technical report [20].

2. Case Study: Stateful Authorization by Example

As a canonical example of stateful authorization policies, we introduce our case study: U.S. policies for access to sensitive intelligence information. The U.S. intelligence community, a cooperative federation of 16 U.S. government agencies, produces classified

documents, which, although inaccessible to the general population, must be accessible to authorized individuals both within the intelligence community and outside it. Our case study formalizes the policies mandated for such access. Our primary source of policies is a set of interviews of intelligence personnel conducted by Brian Witten and Denis Serenyi of Symantec Corporation. Some parts of the policies are based on Executive Orders of the White House [29, 30] or Director of Central Intelligence Directives (DCIDs) [27, 28]. The policies presented here are themselves unclassified, and do not necessarily represent current official practices.

For the purpose of formalization, we assume that the unit of sensitive information is a digital file that is already classified or will potentially be classified. Two facts about sensitive files are of interest to us. First, any sensitive file goes through a life cycle consisting of up to four stages that are shown in Figure 1. Access to a file depends on the stage of the file at the time of access. Second, transitions between stages are dictated by non-mechanizable factors such as human intent and beliefs and are, therefore, not prescribed by the authorization policy itself. As far as the authorization policy is concerned, the stages can change unpredictably. Hence, for the purposes of formalization and enforcement, it is natural to represent the stage of a file as an element of system state external to the policy itself.

Overview of file stages. We briefly describe the stages of a sensitive file. Every newly created official file in an intelligence agency starts in a default stage, in which it is accessible only to the creator (owner) of the file. A file in the default stage may eventually either be deleted by the author, or its content may mature to something publishable, in which case it is promoted, at the discretion of the author, to a working paper. A working paper is more collaborative; access is provided at the discretion of the owner (a group of individuals or an intelligence agency). A file remains a working paper for at most 90 days, after which it must be either deleted, put back in the same stage for 90 more days, classified or declassified. The last two happen only after the content of a working paper is stable. Classifying or declassifying a working paper involves opinion of both the authors and a trained individual called an original classification authority (OCA). If the file is declassified, it is accessible to anyone. Access to a classified file is subject to complicated rules. At the time of classification, a file's secrecy level (confidential, secret, or topsecret) is determined, as are its compartments, which are project-specific classifications that further restrict access. Access to a classified file is based on these attributes as well as others such as the citizenship and employment status of the individual accessing the file. Every classified file is automatically declassified after the lapse of a certain interval of time that is stipulated when the file is classified.

Thus, allowed access to a file varies by stage and transitions between stages involve human factors that are not stipulated by the policy itself, so the policy is stateful in the sense mentioned in the introduction. For our formalization, we assume that the stage of a file is represented as an extended attribute (file meta-data) called status, which is stored in the file system with the file. Our formalization of the policy relies on this extended attribute to determine who may access the file. The relation between the possible values of status and allowed access is summarized in Figure 2.

2.1. Formalizing State in BL

System state is represented in BL formulas through a special class of predicates called state predicates. State predicates can be established in a proof through an external pro-

Value of extended attribute status on file F	Meaning	Who has access
default	F is in default stage	Owner
working T	F is a working paper, put into that stage at time T	Anyone, at the discretion of owner
classified T T'	F is classified from time T to time T'	Complex rules to decide access
declassified	F is declassified	Everyone

Figure 2. Formalization of file stages and permissions allowed in them

cedure, which may vary by application. For formalizing the policy at hand, we need two state predicates: (a) (**has_xattr** f a v), which means that file f has extended attribute a set to value v; in particular, the value of a special attribute status determines the stage of a sensitive file, and (b) (**owner** f k), which means that file f is owned by principal k. We write state predicates in **boldface** to distinguish them from other predicates that are defined by the logical theory and use juxtaposition for applying arguments to predicates, e.g., we write (**owner** f k) instead of the more conventional **owner**(f, k). The letters f, a, v, k range over terms of types file, attribute, attribute value, and principal, respectively. Following standard logic programming convention, we also use the uppercase letters F, A, V, K for implicitly quantified variables of the corresponding types. (This convention is explained further in Section 2.2.) We do not stipulate what principals are — they may be individuals, agencies, or groups. We assume that a procedure to check both **has_xattr** and **owner** in the file system being used for implementation is available.

The attribute status in our formalization can take four possible values corresponding to the four stages in Figure 1. These are listed in Figure 2, together with a description of principals who have access to the file in each stage. Technically, the words default, working, classified, and declassified are uninterpreted function symbols in BL having arities 0, 1, 2, and 0, respectively. The arguments T, T' are variables ranging over time points (discussed later).

2.2. Formalizing Policy Rules in BL

Authorization is formalized in the logic as a predicate (may k f p), which means that principal k has permission p on file f. (may k f p) is *not* a state predicate because it is defined by logical formulas, which are part of our formalization of the policy. Representative examples of formulas defining (may k f p) are shown below:

admin claims (may K F read :-
         has_xattr F status default,
         owner F K) $\circ$ $[-\infty, +\infty]$.
admin claims ((may K F read :-
         has_xattr F status (classified T T'),
         indi/has-clearances/file K F,
         owner F K',
         K' says (may K F read)) @ $[T, T']$) $\circ$ $[-\infty, +\infty]$.
admin claims ((may K F read :-
         has_xattr F status (classified T T')) @ $[T', +\infty]$) $\circ$ $[-\infty, +\infty]$.

admin claims ((may K F read :-

                     has_xattr F status (working T),

                     owner F K',

                     K' says (may K F read),

                     $T' = (T + 90d)$ @ $[T, T']$) $\circ$ $[-\infty, +\infty]$.

The notation s :- $s_1, \ldots, s_n$ means "formula s holds if formulas $s_1, \ldots, s_n$ hold", and is equal to $(s_1 \wedge \ldots \wedge s_n) \supset s$. Uppercase variables like K, F and T occurring in $s, s_1, \ldots, s_n$ are implicitly assumed to be universally quantified outside this formula. The prefix admin claims ... before each rule means that the rule is created by the principal admin, who is assumed to be the ultimate authority on access decisions. The prefix is formally discussed in Section 3. The suffix $\circ$ $[-\infty, +\infty]$ at the end of each rule means that each rule is valid forever.

    The first rule above states that (it is the admin's policy that) principal K may read file F if F is in stage default and K owns F. The latter two facts are determined by looking at the file's meta-data through the external procedure for verifying state predicates.

    The second rule illustrates two central, but possibly unfamiliar, connectives of authorization logics including BL: k says s and s @ $[u_1, u_2]$. The former connective, says, is a principal-indexed modality and has been studied extensively in the context of authorization logics [1–3, 18]. It means that principal k supports formula s or declares that the formula is true. k says s can be established *a priori* (i.e., without a proof) through a digital certificate that contains formula s and is signed by principal k's private key. In fact, this is the *only* way to establish a priori any formula other than a state predicate, constraint (constraints are discussed below), or tautology. s @ $[u_1, u_2]$ captures real time in the logic; it means that formula s holds during the time interval $[u_1, u_2]$, but does not say anything about s outside this interval. The second rule above states that principal K may read file F if F is classified from time T to time T' (**has_xattr** F status (classified T T')), K has the right clearances to read file F (indi/has-clearances/file K F), and the owner K' of file F allows K to read the file (K' says (may K F read)). The suffix @ $[T, T']$ after the formula means that the entire formula and, hence, its consequence (may K F read) apply only in the interval $[T, T']$. Beyond T' the file is effectively declassified and accessible to everyone as captured in the third rule above, which means that during the interval $[T', +\infty]$, any principal K may read a file F that was classified from time T to time T'.

    Our fourth rule highlights the need for another integral feature of BL — constraints, such as the atom $T' = (T + 90d)$. Constraints are similar to state predicates in that they are verified by an external procedure but different in that they are independent of state. Constraints are useful for reasoning about time. For instance, the fourth rule means that principal K may read file F if F is a working paper, the owner K' of F allows the access, and less than 90 days have elapsed since the file became a working paper. The last condition enforces the previously mentioned policy mandate that a file can remain a working paper for at most 90 days.

    The second, third, and fourth rules above also exemplify an interesting interaction between state and time: They apply over time intervals that depend on time values (denoted T in the rules) obtained through state predicates. There are other interesting interactions between state and time that do not occur in our case study but are described in Section 3.3.

    Most of the remaining rules in our case study define the predicate (indi/has-clearances/file K F) from the second rule above. This predicate relates

credentials of an individual *K* and attributes of a classified file *F* to determine whether or not *K* should have access to *F*. Since the rules defining this predicate do not highlight any new features of the logic, we postpone their discussion to Section 6.

3. The Logic BL: Syntax and Proof Theory

Having discussed the features of BL briefly, we now present the logic formally. To keep the description accessible we stage the presentation into three steps, adding more features to the logic at each step. In the first step of presentation (Section 3.1), we consider the core of BL, sorted first-order intuitionistic logic, with the principal-indexed modality k says s. We call this sub-logic BL_0. In Section 3.2, we add state predicates, calling the logic BL_1. Finally, we add explicit time through the connective $s \ @ \ [u_1, u_2]$ to obtain the full logic BL (Section 3.3). We omit a description of the connectives for disjunction and existential quantification from this paper; their details may be found in the first author's thesis [17, Chapters 3 & 4]. Neither connective presents adds any new technical challenges for the proof theory or the metatheory of BL.

3.1. BL_0: First-order Logic and says Modality

The first fragment of BL we consider, BL_0, has the following syntax:

Sorts	σ ::=	principal | time | ...
Terms	t, u, k ::=	Alice | Bob | admin | ...
Predicates	P ::=	may | ...
Atoms	p ::=	$P \ t_1 \ldots t_n$
Formulas	r, s ::=	$p \mid s_1 \wedge s_2 \mid s_1 \supset s_2 \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid k$ says s

Sorts are types for terms. We stipulate at least two sorts: that of principals, principal, and that of time points, time (time points have a special significance in the full logic BL, as discussed in Section 3.3). Although we do not stipulate the domain of terms, it must include at least principals who are authorized access and who create policies (Alice, Bob, admin, etc.). We also allow uninterpreted function symbols in terms. As mentioned earlier, we use the letter k to denote terms of sort principal, f for files, and u for terms of sort time. The letter t denotes a generic term.

    Uninterpreted predicates P allowed in BL_0 are to be distinguished from state predicates that we have been writing in boldface so far. The latter are introduced in Section 3.2. Formulas of BL_0 are either atomic, p, or built from the usual connectives of first-order logic — $\wedge$ (conjunction), $\supset$ (implication), $\top$ (truth), $\bot$ (falsity), $\forall$ (universal quantification) — or, the principal-indexed modality k says s. As explained and illustrated in Section 2, k says s means that principal k supports or states formula s, without necessarily implying that s is true. We often elide the sort σ from the universal quantifier $\forall x{:}\sigma.s$ when it is clear from the context.

    As is standard in intuitionistic logic, negation, $\neg s$, may be defined as $s \supset \bot$. However, because our logic is intuitionistic, $s \vee \neg s$ does not hold for every s, nor is proof by contradiction a valid method of deduction. It has been argued in prior work [18] that absence of these features is beneficial for authorization logics because it forces that evidence of authorization (proofs) be direct. Further, intuitionism forces policy authors to

explicitly specify in their policies the conditions under which access is allowed; the dual approach of listing conditions under which access is disallowed does not work in intuitionistic logic. A significant amount of recent work on enforcement of authorization policies is based on intuitionistic logic (e.g., [1, 4, 23]). Following the same trend, we work with an intuitionistic logic, instead of a classical one.

Proof theory of BL_0. When using authorization logics in practice, access is granted only if there is a proof which justifies the access. Therefore, to understand the meaning of a proposition in authorization logic, we must understand how it can be proved. This naturally leads us to the proof theory of BL_0, i.e., a systematic study of its formal proofs. We adopt Gentzen's sequent calculus style [21] in describing the proof theory and follow Martin-Löf's judgmental approach [26], which has been used previously to describe other modal logics [18, 32]. Briefly, a judgment J is an assertion that can be established through proofs. For BL_0 we need two kinds of judgments: s true, meaning that formula s is true, and k claims s, meaning that principal k claims or supports formula s (but s may or may not be true). The latter is needed to define the meaning of the says modality. A sequent has the following form, where Γ abbreviates a multi-set $J_1, \ldots, J_n$ of judgments and Σ is a finite map from first-order variables free in Γ, s, and k to their sorts.

$$\Sigma; \Gamma \xrightarrow{k} s \text{ true}$$

The informal meaning of the sequent is:

> *Parametrically in the variables in Σ, assuming that everything that principal k claims is true, the judgment s true follows from the judgments in Γ.*

The principal k is called the *view* of the sequent, and can be roughly thought of as the principal relative to whose statements we wish to prove the sequent (hence the hypothesis "assuming that everything that principal k claims is true . . ." in the meaning of the sequent).

Example 3.1. Recall from Section 2.2 that principal admin is the ultimate authority on access. Hence, to allow any access, we must prove the corresponding authorization in the view of admin. For example, to allow Alice to read file secret.txt, we must prove that $\Sigma; \Gamma \xrightarrow{\text{admin}} (\text{may Alice secret.txt read})$ true, where Γ is the set of all rules defining the policy.

Inference Rules. The sequent calculus for BL_0 consists of several inference rules for establishing sequents. Our first rule, (init), is standard and states that if atom p is assumed as a hypothesis, then it can be concluded. We restrict the rule to atoms, but it can be proved that a generalization to arbitrary formulas holds (we prove a similar theorem for the entire logic BL in Section 4). The second rule, (claims), captures the meaning of the view of a sequent: if k claims s is assumed and the view is k, then s true can also be assumed.

$$\frac{}{\Sigma; \Gamma, p \text{ true} \xrightarrow{k} p \text{ true}} \text{init} \qquad \frac{\Sigma; \Gamma, k \text{ claims } s, s \text{ true} \xrightarrow{k} r \text{ true}}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{k} r \text{ true}} \text{claims}$$

Other inference rules of the sequent calculus are directed by connectives. We list below rules for the says connective. The notation $\Gamma\!\downarrow$ in (saysR) denotes the subset of Γ contain-

ing only judgments of the form k' claims s', i.e., $\Gamma\downarrow = \{(k'$ claims $s') \mid k'$ claims $s' \in \Gamma\}$. The rule (saysR) may be read as follows: We can establish that k says s is true in hypotheses Γ in any view k_0 (conclusion of the rule) if we can prove only from $\Gamma\downarrow$ in view k that s is true (premise). Hypotheses of the form s' true are removed from the premise because they may have been introduced in Γ in the view k_0, but may not be claimed or trusted by k, as illustrated at the end of Example 3.2. The second rule (saysL) states that the judgment $(k$ says $s)$ true entails the judgment k claims s. In fact, the two judgments are equivalent in BL_0. (Technically, we say that the connective says internalizes the judgment claims into the syntax of formulas.)

$$\frac{\Sigma;\Gamma\downarrow \xrightarrow{k} s \text{ true}}{\Sigma;\Gamma \xrightarrow{k_0} (k \text{ says } s) \text{ true}}\text{saysR} \qquad \frac{\Sigma;\Gamma, k \text{ claims } s \xrightarrow{k_0} r \text{ true}}{\Sigma;\Gamma, (k \text{ says } s) \text{ true} \xrightarrow{k_0} r \text{ true}}\text{saysL}$$

This interpretation of the says connective using views is novel, and has been chosen to allow for representation of a specific form of delegation of authority, as discussed in Section 5.4.

Rules for connectives of first-order logic ($\wedge$, $\supset$, $\top$, $\bot$, $\forall$) have their standard form, with the exception that the view passes unchanged from the premises to the conclusion. We refer the reader to existing work for details [32]. As an example, we list below the rules for conjunction ($\wedge$).

$$\frac{\Sigma;\Gamma \xrightarrow{k} s_1 \text{ true} \qquad \Sigma;\Gamma \xrightarrow{k} s_2 \text{ true}}{\Sigma;\Gamma \xrightarrow{k} (s_1 \wedge s_2) \text{ true}}\wedge\text{R} \qquad \frac{\Sigma;\Gamma, s_1 \text{ true}, s_2 \text{ true} \xrightarrow{k} r \text{ true}}{\Sigma;\Gamma, (s_1 \wedge s_2) \text{ true} \xrightarrow{k} r \text{ true}}\wedge\text{L}$$

Several standard metatheoretic properties including admissibility of cut and consistency hold of BL_0's proof theory but we refrain from presenting them here because we present similar properties for the larger logic BL in Section 4.

Example 3.2 (Rules for the connective says). We illustrate the rules for the connective says. Consider the sequent $\cdot; (k \text{ says } p) \text{ true} \xrightarrow{k} (k' \text{ says } p) \text{ true}$, where k and k' are distinct ground principals and p is an atom. In words, this sequent states that from the assumption k says p we can deduce k' says p assuming that every statement of principal k is true (because the view here is k). Intuitively, this sequent should not have a proof in the logic because no relation between statements of k and those of k' has been assumed. We justify this intuition formally by trying to construct a proof of the sequent reasoning backwards using the rules of calculus and explaining why every such attempt fails. (We drop the empty context Σ from every sequent to reduce clutter.)

To construct a proof of $(k \text{ says } p) \text{ true} \xrightarrow{k} (k' \text{ says } p) \text{ true}$, we can either try to apply the (saysR) rule or the (saysL) rule. Trying the former, we reduce to proving that $((k \text{ says } p) \text{ true})\downarrow \xrightarrow{k'} p \text{ true}$, i.e., $\cdot \xrightarrow{k} p \text{ true}$, to which no other rule applies, so this attempt fails. If we instead try the rule (saysL), we reduce to proving $(k \text{ says } p) \text{ true}, k \text{ claims } p \xrightarrow{k}$ $(k' \text{ says } p) \text{ true}$. Now we can apply either the rule (claims) or the rule (saysR). We show here the former case, which is more interesting (the latter case fails as well). Using the rule (claims), it suffices to prove that $(k \text{ says } p) \text{ true}, k \text{ claims } p, p \text{ true} \xrightarrow{k} (k' \text{ says } p) \text{ true}$. Now we are forced to use (saysR), which reduces our goal to $((k \text{ says } p) \text{ true}, k \text{ claims}$

p, p true)$\downarrow \xrightarrow{k'} p$ true. Observe that $((k \text{ says } p) \text{ true}, k \text{ claims } p, p \text{ true})\downarrow = k \text{ claims } p$, so we need to show that $k \text{ claims } p \xrightarrow{k'} p$ true. No rule applies now, so the sequent is unprovable, as expected.

Note that if in the rule (saysR) we were to allow all of Γ in the premise instead of $\Gamma\downarrow$, then this (intuitively unprovable) sequent would have a proof in BL_0 because after the last application of (saysR) in the above discussion, we would have ended up with the sequent $(k \text{ says } p) \text{ true}, k \text{ claims } p, p \text{ true} \xrightarrow{k'} p$ true, which has a one-step proof by the rule (init). In other words, in applying the (saysR) rule, we would have changed the view from k in the conclusion to k' in the premise and yet, unjustifiably, retained in the premise the assumption p true, which was obtained from the statement $k \text{ says } p$ made by principal k, not k'. By removing hypotheses of the form s' true in the premise of (saysR) we avoid this possibility and ensure that statements of distinct principals are not confused in the logic a priori.

3.2. BL_1: State Predicates

To represent stateful policies, examples of which were shown in Section 2, we extend BL_0 with a special class of atomic formulas called stateful atoms, denoted i, and add a new form of hypotheses — a set of stateful atoms, E — to sequents. The resulting logic, BL_1, has sequents of the form $\Sigma; E; \Gamma \xrightarrow{k} s$ true, which informally mean that:

> *Parametrically in the variables in Σ, assuming that everything that principal k claims is true, the judgment s true follows from the judgments in Γ in any environment that validates all stateful atoms in E.*

In practice, stateful atoms in E may be discharged by an external procedure that has access to the environment or system state.

Syntax. The syntax of BL_1 formulas is shown below. The meta-variables p and t inherit their syntax from BL_0. State predicates I are assumed to be distinct from uninterpreted predicates P.

State predicates $\quad I ::= \textbf{has\_xattr} \mid \textbf{owner} \mid \ldots$
Stateful atoms $\quad i ::= I \, t_1 \ldots t_n$
Formulas $\quad r, s ::= p \mid i \mid s_1 \wedge s_2 \mid s_1 \supset s_2 \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid k \text{ says } s$

Example 3.3. In BL_1, Alice is allowed to read file secret.txt iff there is a proof of $\Sigma; E; \Gamma \xrightarrow{\text{admin}} (\text{may Alice secret.txt read})$ true, where Γ is the set of all rules defining the policy and E is a set of stateful atoms that hold in the environment prevailing at the time of access.

Proof Theory of BL_1. We incorporate relations between stateful atoms into the proof theory through an abstract judgment $\Sigma; E \models i$, which means that "for all ground instances of variables in Σ, any environment that satisfies all stateful atoms in E also satisfies atom i". We do not stipulate any rules to establish this judgment since they may vary from environment to environment. For instance:

- If names a.txt and b.txt alias the same file, then $\Sigma, a, v; (\textbf{has\_xattr } \text{a.txt } a \, v) \models (\textbf{has\_xattr } \text{b.txt } a \, v)$.

- If files inherit owners from containing directories, then $\Sigma, k; (\textbf{owner} \ (/\textsf{usr}) \ k) \models (\textbf{owner} \ (/\textsf{usr}/\textsf{a.txt}) \ k)$.

In the simplest instance, the judgment $\Sigma; E \models i$ may hold if and only if $i \in E$. Our metatheoretic results (Section 4) assume only the following properties of this judgment, all of which follow from its intuitive explanation. $\Sigma \vdash t : \sigma$ means that term t has sort σ given the sorting Σ for variables.

$\Sigma; E, i \models i$ (Identity)

$\Sigma; E \models i$ implies both $\Sigma, x{:}\sigma; E \models i$ and $\Sigma; E, E' \models i$ (Weakening)

$\Sigma; E \models i$ and $\Sigma; E, i \models i'$ imply $\Sigma; E \models i'$ (Cut)

$\Sigma, x{:}\sigma; E \models i$ implies $\Sigma; E[t/x] \models i[t/x]$ if $\mathtt{fv}(t) \subseteq \Sigma$ and $\Sigma \vdash t : \sigma$ (Substitution)

As explained earlier, BL_1 sequents have the form $\Sigma; E; \Gamma \xrightarrow{k} s \ \textsf{true}$. BL_1 inherits all inference rules of BL_0 with the proviso that the new context E passes unchanged from the conclusion to premises in all rules. We do not reiterate these rules. Two new rules for reasoning about stateful atoms are added. The first rule states that the judgment $i \ \textsf{true}$ holds if $E \models i$ for the assumed state E. The second rule means that a hypothesis $i \ \textsf{true}$ implies that the stateful atom i holds. Together, the two rules imply that the judgment $i \ \textsf{true}$ is equivalent to the atom i holding in the prevailing environment E, which closely couples the stateful *formula* i to its intended interpretation.

$$\frac{\Sigma; E \models i}{\Sigma; E; \Gamma \xrightarrow{k} i \ \textsf{true}} \textsf{stateR} \qquad\qquad \frac{\Sigma; E, i; \Gamma \xrightarrow{k} s \ \textsf{true}}{\Sigma; E; \Gamma, i \ \textsf{true} \xrightarrow{k} s \ \textsf{true}} \textsf{stateL}$$

We list below admissible and inadmissible statements relating to stateful atoms and the says connective. The second and third statements mean that a false stateful atom signed by a principal does not contaminate the entire logic.

$\vdash i \supset k \ \textsf{says} \ i$

$\nvdash (k \ \textsf{says} \ i) \supset i$

$\nvdash (k \ \textsf{says} \ i) \supset (k' \ \textsf{says} \ i)$ if $k \neq k'$

BL_1's proof theory satisfies several standard metatheoretic properties including admissibility of cut and consistency but we refrain from presenting them here because we present similar properties for the larger logic BL in Section 4.

3.3. BL: Explicit Time and @ Connective

Finally, we add explicit time to the logic by including the connective $s \ @ \ [u_1, u_2]$. This connective is based on our prior work with DeYoung for a different logic η [14], which did not include state. The reason for considering the extension with time is two-fold. First, explicit time is needed to correctly represent policy rules that have a pre-determined expiration, as well as other rules that limit the temporal validity of formulas (e.g., the second, third, and fourth rules of Section 2). Second, there are important design decisions in the interaction between state and time that we wish to highlight. In particular, due to this interaction the obvious interpretation of stateful atoms allows expressing unenforceable policies in BL, so we use a somewhat non-intuitive interpretation of stateful atoms in BL, as explained in Section 3.3.1.

Since s @ $[u_1, u_2]$ means that s holds throughout the interval $[u_1, u_2]$, it seems reasonable that s @ $[u_1, u_2]$ imply s @ $[u'_1, u'_2]$ if $u_1 \le u'_1$ and $u'_2 \le u_2$. To make such properties admissible in the logic, we need a theory of the total order $u_1 \le u_2$ on time points and, for expressing certain policies (e.g., the fourth rule in Section 2), we also need a theory of arithmetic over time points. We include both by adding a single *constraint domain* of time points to the logic. From the perspective of proof theory, constraints are similar to state. However, the external procedure for verifying constraints does not depend on state.

Syntax. Time points are either integers or one of the elements $\{-\infty, +\infty\}$. The numbers represent time elapsed in seconds from a fixed point of reference. In the concrete syntax, we often write absolute time points in the format YYYY:MM:DD:hh:mm:ss. We include an infix function symbol $+$ of arity 2. A new syntactic class of atomic formulas called constraints, denoted c, is also added. Constraints are predicates of one of two forms: $u_1 \le u_2$ and $u_1 = u_2$.

$$
\begin{array}{lll}
\text{Terms} & t, u, k ::= & \mathsf{Alice} \mid \mathsf{Bob} \mid \text{YYYY:MM:DD:hh:mm:ss} \mid -\infty \mid +\infty \mid \\
& & u_1 + u_2 \mid \ldots \\
\text{Constraints} & c ::= & u_1 \le u_2 \mid u_1 = u_2 \\
\text{Formulas} & r, s ::= & p \mid i \mid c \mid s_1 \wedge s_2 \mid s_1 \supset s_2 \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid k \;\mathsf{says}\; s \mid \\
& & s \; @ \; [u_1, u_2]
\end{array}
$$

3.3.1. Proof Theory of BL

The addition of time requires a significant change to the logic's judgments [14]. Instead of the judgments s true and k claims s, we use refined judgments $s \circ [u_1, u_2]$ (s is true throughout the interval $[u_1, u_2]$) and k claims $s \circ [u_1, u_2]$ (k claims that s is true throughout the interval $[u_1, u_2]$). Sequents in BL have the form $\Sigma; \Psi; E; \Gamma \xrightarrow{k, u_1, u_2} s \circ [u'_1, u'_2]$. Here, Ψ is a set of constraints. The meaning of the sequent is:

Parametrically in the variables in Σ, assuming that everything that principal k claims about intervals containing $[u_1, u_2]$ is true, the judgment $s \circ [u'_1, u'_2]$ follows from the judgments in Γ in any environment that validates all stateful atoms in E, if all constraints in Ψ hold.

Besides the addition of constraints as hypotheses, another change is the addition of an interval of time to the view. This is not particularly important since we could also have constructed a logic without time intervals in views (for details of the trade-offs involved in making this choice, see [17, Section 4.4]).

Relations between constraints are incorporated into the logic through an abstract judgment $\Sigma; \Psi \models c$, which is similar to $\Sigma; E \models i$. As for the latter judgment, our metatheoretic properties rely only on basic properties of $\Sigma; \Psi \models c$. These properties include analogues of properties (Identity) – (Substitution) of Section 3.2. In addition, we require that $\le$ be reflexive and transitive.

$$
\begin{array}{ll}
\Sigma; \Psi \models u \le u & \text{(Reflexivity)} \\
\Sigma; \Psi \models u_1 \le u_2 \text{ and } \Sigma; \Psi \models u_2 \le u_3 \text{ imply } \Sigma; \Psi \models u_1 \le u_3 & \text{(Transitivity)}
\end{array}
$$

Inference rules of the sequent calculus for BL are derived from those of BL_1, taking into account carefully the interaction between time and the different connectives. This interaction is non-trivial in most cases, as was observed in prior work [14].

Basic rules. The following two rules relate the judgments of BL. Rule (init) generalizes its homonym from BL_0: If we assume that atom p holds throughout the interval $[u'_1, u'_2]$, then we can conclude that p holds throughout a *subinterval* $[u_1, u_2]$, i.e., if $u'_1 \leq u_1$ and $u_2 \leq u'_2$. Rule (claims) allows us to promote the hypothesis k claims $s \circ [u_1, u_2]$ to $s \circ [u_1, u_2]$ if the principal in the view v is k, and the interval $[u_b, u_e]$ in v is a subinterval of $[u_1, u_2]$.

$$\frac{\Sigma;\Psi \models u'_1 \leq u_1 \qquad \Sigma;\Psi \models u_2 \leq u'_2}{\Sigma;\Psi;E;\Gamma, p \circ [u'_1, u'_2] \xrightarrow{v} p \circ [u_1, u_2]}\text{init}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Gamma, k \text{ claims } s \circ [u_1, u_2], s \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2] \\ v = k, u_b, u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2\end{array}}{\Sigma;\Psi;E;\Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2]}\text{claims}$$

Connective @. As in prior work [14], $s @ [u_1, u_2]$ internalizes the judgment $s \circ [u_1, u_2]$ into the syntax of formulas. Because $s @ [u_1, u_2]$ means that s holds throughout $[u_1, u_2]$, a further qualification by adding $\circ [u'_1, u'_2]$ as in $(s @ [u_1, u_2]) \circ [u'_1, u'_2]$ does not add anything to the meaning, so the judgments $s \circ [u_1, u_2]$ and $s @ [u_1, u_2] \circ [u'_1, u'_2]$ are equivalent. As a result, we can replace $(s @ [u_1, u_2]) \circ [u'_1, u'_2]$ with $s \circ [u_1, u_2]$ in the both the hypotheses and the conclusion of a sequent, which is precisely what the following two rules for the @ connective allow. (Here, v denotes an arbitrary view.)

$$\frac{\Sigma;\Psi;E;\Gamma \xrightarrow{v} s \circ [u_1, u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (s @ [u_1, u_2]) \circ [u'_1, u'_2]}\text{@R} \qquad \frac{\Sigma;\Psi;E;\Gamma, s \circ [u_1, u_2] \xrightarrow{v} r \circ [u''_1, u''_2]}{\Sigma;\Psi;E;\Gamma, (s @ [u_1, u_2]) \circ [u'_1, u'_2] \xrightarrow{v} r \circ [u''_1, u''_2]}\text{@L}$$

State predicates. If i is a stateful atom, what should $i \circ [u_1, u_2]$ mean? One possibility (which we don't use in BL) is to apply the usual meaning of $s \circ [u_1, u_2]$, namely that the stateful atom i holds throughout the time interval $[u_1, u_2]$. Although intuitive, this interpretation can result in policies that are impossible to enforce. Consider, for example, the policy $(i @ [T, T+5]) \supset ((\text{may } K F \text{ read}) @ [T, T])$. Intuitively, the policy says that a principal K may read file F at time T if i holds in the interval $[T, T+5]$. Thus, permission to access file F at time T refers to *state at later points of time*, which is, of course, impossible to enforce in a reference monitor.

To avoid such non-enforceable policies, we make a substantial design decision in BL: We assume that all stateful atoms are interpreted at exactly one point of time and $i \circ [u_1, u_2]$ simply means that i holds in the environment at this point of time (independent of u_1 and u_2). The logic does not stipulate what that point of time is, but it seems practical to use the time at which the access happens. In that interpretation, $i \circ [u_1, u_2]$ means that i holds at the time of access. Following this decision, the following rules for stateful atoms are self-explanatory:

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} i \circ [u_1, u_2]}\text{stateR} \qquad \frac{\Sigma;\Psi;E,i;\Gamma \xrightarrow{v} r \circ [u'_1, u'_2]}{\Sigma;\Psi;E;\Gamma, i \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2]}\text{stateL}$$

Seemingly, we are limiting the logic's expressiveness because we are eliminating (enforceable) policy rules that refer to stateful atoms in intervals prior to access. Such policy

rules occur rarely; in particular, no such rule occurs in our entire case study. Further, even when they occur, such policy rules can be encoded by requiring evidence of the stateful atom(s) having been true in the past (e.g., a trusted observer's certificate) to exist at the time of access. As a result, we consider our design reasonable.

Constraints. Constraints and state predicates have similar treatments in the proof theory. Consequently, the rules governing constraints are:

$$\frac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Gamma \xrightarrow{v} c \circ [u_1, u_2]} \text{consR} \qquad \frac{\Sigma; \Psi, c; E; \Gamma \xrightarrow{v} r \circ [u'_1, u'_2]}{\Sigma; \Psi; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2]} \text{consL}$$

As opposed to other formal systems [22, 33], we do not allow the deduction of logical falsity $\perp$ and, hence, every formula, when constraints Ψ are contradictory. The absence of such deduction is desirable for an authorization logic because it prevents implicit authorizations, as explained in Section 5.2.

Connective says. The rules for says are similar to the rules for the same connective in BL_0, but also take into account time intervals. For example, rule (saysR) states that $(k$ says $s) \circ [u_1, u_2]$ can be established in any view v from hypotheses Γ if from the hypotheses $\Gamma\downarrow$ (the subset of Γ containing assumptions of the form k' claims $s' \circ [u'_1, u'_2]$), we can prove $s \circ [u_1, u_2]$ in the view k, u_1, u_2. Rule (saysL) relies on the fact that $(k$ says $s)$ claims $[u_1, u_2]$ and k claims $s \circ [u_1, u_2]$ are equivalent judgments in BL.

$$\frac{\Sigma; \Psi; E; \Gamma\downarrow \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{v} (k \text{ says } s) \circ [u_1, u_2]} \text{saysR}$$

$$\frac{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2]}{\Sigma; \Psi; E; \Gamma, (k \text{ says } s) \circ [u_1, u_2] \xrightarrow{v} r \circ [u'_1, u'_2]} \text{saysL}$$

Connective $\supset$. Based on prior work [14], a proof of $(s_1 \supset s_2) \circ [u_1, u_2]$ is a method of converting a proof of $s_1 \circ [u'_1, u'_2]$ to a proof of $s_2 \circ [u'_1, u'_2]$ for any subinterval $[u'_1, u'_2]$ of $[u_1, u_2]$. Accordingly, the rule $(\supset R)$ means that $(s_1 \supset s_2) \circ [u_1, u_2]$ is provable (conclusion) if for fresh variables x_1 and x_2, $s_2 \circ [x_1, x_2]$ is provable from the assumptions $[x_1, x_2] \subseteq [u_1, u_2]$ and $s_1 \circ [x_1, x_2]$ (premise). Dually, the rule $(\supset L)$ allows us to assume $s_2 \circ [u'_1, u'_2]$ (second premise) if $s_1 \circ [u'_1, u'_2]$ is provable (first premise), $[u'_1, u'_2] \subseteq [u_1, u_2]$ (third and fourth premises), and $(s_1 \supset s_2) \circ [u_1, u_2]$ has been assumed (hypotheses of conclusion).

$$\frac{\Sigma, x_1:\text{time}, x_2:\text{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{v} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{v} (s_1 \supset s_2) \circ [u_1, u_2]} \supset R$$

$$\frac{\begin{array}{c} \Sigma; \Psi; E; \Gamma, (s_1 \supset s_2) \circ [u_1, u_2] \xrightarrow{v} s_1 \circ [u'_1, u'_2] \\ \Sigma; \Psi; E; \Gamma, (s_1 \supset s_2) \circ [u_1, u_2], s_2 \circ [u'_1, u'_2] \xrightarrow{v} r \circ [u''_1, u''_2] \\ \Sigma; \Psi \models u_1 \leq u'_1 \qquad \Sigma; \Psi \models u'_2 \leq u_2 \end{array}}{\Sigma; \Psi; E; \Gamma, (s_1 \supset s_2) \circ [u_1, u_2] \xrightarrow{v} r \circ [u''_1, u''_2]} \supset L$$

Other connectives. Other connectives of BL ($\forall$, $\wedge$, $\top$, $\perp$) do not have a significant interaction with time because these connectives commute with the @ connective. This results in straightforward inference rules that are shown in Appendix A.

Disjunctive reasoning about intervals of time. BL's proof system does not, in general, allow combining proofs of $s \circ [u_1, u_2]$ and $s \circ [u_2, u_3]$ into a single proof of $s \circ [u_1, u_3]$ when $[u_1, u_2]$ and $[u_2, u_3]$ are overlapping intervals. The reason for this decision is to keep automatic proof search tractable: If we were to allow such combinations in the proof system, then a procedure to find a proof of $s \circ [u_1, u_3]$ would, in general, have try the *infinitely* many choices of intervals $[u_1, u_2]$ and $[u_2, u_3]$ whose union is $[u_1, u_3]$. (A detailed discussion of proof search for BL is beyond the scope of this paper, but we point the reader to [17, Chapter 6] for details.) Whereas the absence of such combinations has theoretical implications, e.g., there are judgments like $s \circ [u_1, u_3]$ in the earlier sentences that may hold intuitively but cannot be proved in the logic, it does not necessarily limit expressiveness in practice. For instance, the formula s @ $[u_1, u_3]$ may be justified in a reference monitor using *two* proofs instead of just one, where the first proof establishes $s \circ [u_1, u_2]$ and the second proof establishes $s \circ [u_2, u_3]$ for some u_2.

3.3.2. Summary

We summarize the proof system of BL by listing below some admissible and some inadmissible properties. $s \equiv r$ denotes $(s \supset r) \wedge (r \supset s)$, $\vdash s$ means that $\Sigma; \cdot; \cdot; \cdot \xrightarrow{v} s \circ [u_1, u_2]$ for all u_1, u_2, v and appropriate Σ, and $\nvdash s$ means that the latter is not true for s in the stated generality.

1. $\vdash ((u_1 \leq u_1') \wedge (u_2' \leq u_2)) \supset ((s$ @ $[u_1, u_2]) \supset (s$ @ $[u_1', u_2']))$
2. $\vdash ((s$ @ $[u_1, u_2])$ @ $[u_1', u_2']) \equiv (s$ @ $[u_1, u_2])$
3. $\vdash ((s_1 \wedge s_2)$ @ $[u_1, u_2]) \equiv ((s_1$ @ $[u_1, u_2]) \wedge (s_2$ @ $[u_1, u_2]))$
4. $\vdash ((\forall x.s)$ @ $[u_1, u_2]) \equiv (\forall x.(s$ @ $[u_1, u_2]))$ if $(x \notin u_1, u_2)$
5. $\vdash \top$ @ $[u_1, u_2]$
6. $\vdash (\perp$ @ $[u_1, u_2]) \supset s$
7. There is no interval $[u_1, u_2]$ such that $\vdash \perp$ @ $[u_1, u_2]$
8. $\vdash ((s_1 \supset s_2)$ @ $[u_1, u_2]) \equiv (\forall x_1.\forall x_2.(((u_1 \leq x_1) \wedge (x_2 \leq u_2) \wedge (s_1$ @ $[x_1, x_2]))$
$\supset (s_2$ @ $[x_1, x_2])))$
9. $\vdash ((k \text{ says } s)$ @ $[u_1, u_2]) \supset (k \text{ says } (s$ @ $[u_1, u_2]))$
10. $\nvdash (k \text{ says } (s$ @ $[u_1, u_2])) \supset ((k \text{ says } s)$ @ $[u_1, u_2])$

Property 1 states that if a formula s is true throughout an interval $[u_1, u_2]$, then it is also true throughout any subinterval $[u_1', u_2']$. Property 2 states that the meaning of a formula with a top-level @ connective, like s @ $[u_1, u_2]$, is not altered by a further qualification with @. Properties 3 and 4 mean that the connective @ commutes with the connectives $\wedge$ and $\forall$. Property 6 means that if $\perp$ holds in *any* interval of time $[u_1, u_2]$, then any formula s is provable. Property 7 states that, without any assumptions, we cannot derive $\perp$ in any interval of time; together with Property 6, this is a proof-theoretic statement of the logic's consistency. Property 8 asserts that $s_1 \supset s_2$ holds throughout the interval $[u_1, u_2]$ iff the truth of s_1 in every subinterval $[x_1, x_2]$ implies the truth of s_2 in the same subinterval. Properties 9 and 10 mean that @ commutes with says in one direction only.

4. Metatheory of BL

We prove several important metatheoretic properties of BL.[3] The first lemma below states that proofs respect substitution of stateful atoms, which, in a sense, means that the proof theory preserves the meaning of the judgment $\Sigma; E \models i$. A similar property holds for constraints, but we do not state it explicitly.

Lemma 4.1 (State substitution). $\Sigma; E \models i$ and $\Sigma; \Psi; E, i; \Gamma \xrightarrow{v} r \circ [u_1, u_2]$ imply $\Sigma; \Psi; E; \Gamma \xrightarrow{v} r \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E, i; \Gamma \xrightarrow{v} r \circ [u_1, u_2]$. For the case of rule (stateR), we rely on the property (Cut) from Section 3.2. □

BL's proof system also satisfies standard metatheoretic properties like closure under weakening of hypotheses and substitution of ground terms for parameters. We do not state these explicitly. The next significant metatheoretic property is *subsumption*, which allows us to make the time intervals in views and conclusions smaller.

Theorem 4.2 (Subsumption). *Suppose* $\Sigma; \Psi \models u_1 \le u_1'$ *and* $\Sigma; \Psi \models u_2' \le u_2$. *Then, the following hold.*

1. *If* $\Sigma; \Psi; E; \Gamma \xrightarrow{k, u_1, u_2} J$, *then* $\Sigma; \Psi; E; \Gamma \xrightarrow{k, u_1', u_2'} J$.
2. *If* $\Sigma; \Psi; E; \Gamma \xrightarrow{v} s \circ [u_1, u_2]$, *then* $\Sigma; \Psi; E; \Gamma \xrightarrow{v} s \circ [u_1', u_2']$.

Proof. (1) follows by induction on the given derivation. (2) also follows by induction on the given derivation, using (1) for the case of rule (saysR). □

Theorem 4.2(2) is relevant in practice because it formalizes the following (intuitive) fact: To authorize an access over an interval $[u_1', u_2']$, it is okay to validate the access over a larger interval $[u_1, u_2]$.

Our main metatheoretic results are the admissibility of cut — that the proof of a judgment can be used to discharge the same judgment used as a hypothesis in another proof — and identity — any judgment assumed as hypothesis can be concluded. Admissibility of cut is a proof-theoretic statement of soundness of a logic. Dually, identity is a proof-theoretic statement of completeness of the logic's inference rules. Together, the proofs of the two theorems show that the rules of the logic fit well with each other [32].

Theorem 4.3 (Admissibility of cut). $\Sigma; \Psi; E; \Gamma \xrightarrow{v} s \circ [u_1, u_2]$ *and* $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{v} s' \circ [u_1', u_2']$ *imply* $\Sigma; \Psi; E; \Gamma \xrightarrow{v} s' \circ [u_1', u_2']$.

Proof. First, we strengthen the theorem by adding a second statement: $\Sigma; \Psi; E; \Gamma \downarrow \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Gamma, k$ claims $s \circ [u_1, u_2] \xrightarrow{v} s' \circ [u_1', u_2']$ imply $\Sigma; \Psi; E; \Gamma \xrightarrow{v} s' \circ [u_1', u_2']$. Both statements are then proved simultaneously by lexicographic induction, first on the structure of s, and then on the depths of the two given derivations, as in prior work [31]. The proof uses Lemma 4.1 and Theorem 4.2. □

---

[3]Proofs of all lemmas and theorems with key induction cases are present in the first author's thesis [17, Chapter 4].

Theorem 4.4 (Identity). $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{v} s \circ [u_1, u_2]$.

Proof. By induction on *s*. □

By an analysis of inference rules, it also follows that the logic is proof-theoretically consistent, i.e., $\bot$ cannot be established a priori (i.e., in an empty context). Similarly, k says $\bot$ cannot be proved a priori.

Theorem 4.5 (Consistency). *(1)* $\Sigma; \cdot; \cdot; \cdot \not\xrightarrow{v} \bot \circ [u_1, u_2]$, *and (2)* $\Sigma; \cdot; \cdot; \cdot \not\xrightarrow{v} (k$ says $\bot) \circ [u_1, u_2]$.

Proof. By exhaustive backwards application of all applicable rules and failing to complete a proof in each case. □

5. Discussion

We end our description of the logic BL by discussing briefly how policies written in it may be enforced, why we do not allow deduction of arbitrary formulas from contradictory constraints, analyzing the possibility of treating time as a special case of state (instead of using the connective @), and commenting on the interpretation of the says connective that we use in BL.

5.1. Enforcement of BL Policies

Policies expressed in BL can be enforced using the standard architecture of proof-carrying authorization (PCA) [3, 5]. PCA combines cryptography and formal proofs for rigorous policy enforcement. A minor novelty in using it for BL is in the treatment of system state. Although a detailed discussion of enforcement of BL policies through PCA is the subject of a separate paper [19], we briefly discuss the main ideas here.

The central tenet of enforcing policies expressed in any logic is to allow an access φ iff the prevailing policy Γ entails it, i.e., $\Gamma \vdash \varphi$. In an implementation, logical judgments constituting the access policy, i.e., the context Γ, may be established by signed certificates. For example, a reference monitor may admit the BL judgment k claims $s \circ [u_1, u_2]$ as a valid policy rule if there exists a certificate containing formula *s*, digitally signed by principal *k* and having validity $[u_1, u_2]$. The judgment φ representing access depends on the policy, but is generally straightforward. For example, if BL is used to represent file access policies, then the judgment φ representing principal *k*'s access to read file *f* at time *u* may be $(\mathtt{may\_read}\ k\ f) \circ [u, u]$.

The new aspect of enforcement for BL is the representation of state on which authorization may depend. This is not difficult to incorporate because state can be represented explicitly in BL's sequents through the context *E*. In particular, a reference monitor may authorize access φ iff there exist *E* and Γ such that for a fresh view *v*:

1. There is a proof of $\cdot; E; \cdot; \Gamma \xrightarrow{v} \varphi$.
2. There are digital certificates to justify each hypothesis in Γ, as described above.
3. All stateful atoms in *E* can be checked on the system state using a given trusted external procedure for their verification.

$$\frac{\Psi = (\texttt{price}(a) = 10) \qquad (\Psi, \texttt{price}(a) > 75) \text{ is contradictory}}{\cfrac{\Sigma, x_1:\text{time}, x_2:\text{time}; \Psi, \texttt{price}(a) > 75; \cdot; \cdot \xrightarrow{v} (\texttt{approves\_manager } a) \circ [x_1, x_2]}{\cfrac{\Sigma, x_1:\text{time}, x_2:\text{time}; \Psi; \cdot; (\texttt{price}(a) > 75) \circ [x_1, x_2] \xrightarrow{v} (\texttt{approves\_manager } a) \circ [x_1, x_2]}{\cfrac{\Sigma; \Psi; \cdot; \cdot \xrightarrow{v} (\texttt{price}(a) > 75) \supset (\texttt{approves\_manager } a)}{\cfrac{\Sigma; \Psi; \cdot; ((\texttt{price}(a) > 75) \supset (\texttt{approves\_manager } a)) \supset (\texttt{purchase } a) \xrightarrow{v} \texttt{purchase } a}{\Sigma; \Psi; \cdot; pol \xrightarrow{v} \texttt{purchase } a} \; \forall L} \; \supset L} \; \supset R} \; \text{consL}}$$

Figure 3. Proof of statement from Section 5.2 in BL's sequent calculus extended with reasoning from contradictory constraints. To simplify notation, we omit the suffix $\circ [-\infty, +\infty]$ from judgments, unnecessary hypothesis, and premises that are trivially provable.

Because the logic BL is undecidable, it is unreasonable to require the reference monitor to find the proof in step (1) and, instead, push the onus of providing E, Γ, the proof of (1) and the certificates of (2) to the principal requesting access. The reference monitor then only needs to *verify* that the proof provided for (1) is correct, that the certificates provided for (2) check cryptographically and that (3) holds, all of which are relatively straightforward. This architecture, which moves the burden of providing evidence to the principal requesting access, is called proof-carrying authorization (PCA). We have implemented a PCA-based file system, PCFS, that can enforce policies written in BL [19]. The implementation includes both a verifier for BL proofs (which is needed for step (1) above) as well as a goal-directed automatic, but incomplete, theorem prover for BL.

5.2. Reasoning from Contradictory Constraints

Some formal systems containing both logical deduction and constraints allow the deduction of $\perp$ and, hence, every formula if assumed constraints Ψ are contradictory [22, 33]. BL does not allow such deduction because it can result in unexpected, implicit consequences of policies. For example, consider the following policy rule which says that if the price of an item x is greater than \$75, then it can be purchased in an office with the manager's approval (the function $\texttt{price}(x)$ returns the price of item x):

$$pol = \forall x. \, (((\texttt{price}(x) > 75) \supset (\texttt{approves\_manager } x)) \supset (\texttt{purchase } x))$$

Note that this policy rule does *not explicitly* list the conditions under which items priced less than \$75 may be purchased; such items may be covered by other policy rule(s). However, if we allow deduction of arbitrary formulas from contradictory constraints then, in the resulting logic, this policy allows the purchase of any item priced less than \$75. For instance, if a is an item with $\texttt{price}(a) = 10$, then we would be able to prove in the resulting logic that $\Sigma; \Psi; \cdot; pol \circ [-\infty, +\infty] \xrightarrow{v} (\texttt{purchase } a) \circ [-\infty, +\infty]$ for $\Sigma = a$, $\Psi = (\texttt{price}(a) = 10)$ and arbitrary v. A sequent calculus proof of this fact is shown in Figure 3. The key step in the proof is the topmost rule in the figure: The contradictory constraints $\texttt{price}(a) = 10$ and $\texttt{price}(a) > 75$ allow the deduction of the formula $(\texttt{approves\_manager } a) \circ [x_1, x_2]$.

Such implicit consequences of policies are clearly undesirable and, to prevent them, BL does not allow deduction of arbitrary formulas from contradictory constraints.

5.3. Time as a Special Case of State?

In BL, time is represented using the connective s @ $[u_1, u_2]$. However, time is inherently an element of system state. This raises the following natural question: Is a representation of time separate from state necessary? Specifically, if we include in our logic an interpreted constant, say localtime, which evaluates to the current time (as in some prior work [5, 8]), and drop the connective @, will we have the same expressiveness as BL? The answer to this question is no; BL is strictly more expressive than a logic without @ but with a constant localtime.

 This difference exists because BL's @ connective allows us to say that a formula holds at a specific point of time (or throughout an interval of time), whereas the constant localtime only allows us to relate the current time to other points of time. Consequently, any policy that refers to the truth of formulas at two different points of time cannot be expressed using localtime alone. For instance, the BL policy rule $(p$ @ $[u, u]) \supset (p'$ @ $[u+5, u+5])$ (if predicate p holds at time u, then p' holds at $u+5$) is impossible to represent in such a setup. Similarly, policy rules that are valid on time intervals that depend on time points written in system state can be represented using @ but cannot be expressed using localtime alone. The second and fourth rules of Section 2.2 are examples of such rules. Consequently, representing time through the @ connective is useful even when state is included in the logic.

5.4. Delegation with BL's says *Connective*

The connective k says s merits additional discussion because its interpretation is BL_0 (and, by inheritance, in BL_1 and BL) is different from that in prior work. Starting with the work of Lampson et al. [24], the says connective has been included in several authorization logics [1, 2, 18]. However, in each case, the inference rules defining says (and, hence, the formal meaning of k says s) are different. For instance, Lampson et al. [24] and Abadi et al. [2] treat $(k$ says $\cdot)$ as a normal necessitation modality of classical modal logic. In more recent papers, Abadi [1] and the authors [18], treat $(k$ says $\cdot)$ as a lax modality [16]. Each treatment has its own merits and demerits in expressing and reasoning about policies. In BL_0, BL_1, and BL we define says with a new and non-standard set of inference rules using views (rules (saysR), (saysL) and (claims) in Section 3.1). This new interpretation has been chosen to allow for accurate representation of a specific form of delegation (transfer of authority) from one principal to another, which, we believe, none of the prior definitions of says allow. In this section, we justify this new choice. We base our discussion on BL_0 instead of BL to keep the notation simple and introduce an axiomatic characterization (Hilbert-style presentation) of BL_0's says connective to facilitate our discussion.

Axiomatic characterization of BL_0. Besides the sequent calculus defined in Section 3.1, BL_0 can be equivalently characterized by taking any axiomatization of first-order intuitionistic logic and adding the following rule and axioms for the connective says. (Standard notation: $\vdash s$ means that s is provable without assumptions.)

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

$$\vdash (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2)) \quad \text{(K)}$$
$$\vdash (k \text{ says } s) \supset k' \text{ says } k \text{ says } s \quad \text{(I)}$$
$$\vdash k \text{ says } ((k \text{ says } s) \supset s) \quad \text{(C)}$$

Theorem 5.1 (Equivalence of proof systems). *The above rule and axioms for* says, *together with any axiomatization of intuitionistic first-order logic, are equivalent to the sequent calculus for BL_0, i.e., for any s and k without free variables, $\vdash s$ if and only if $\cdot; \cdot \xrightarrow{k} s$* true.

Proof. By simulating a proof in the sequent calculus in the axiomatic calculus and viceversa. The technique is standard. For details see [17, Chapter 3]. □

Delegation in BL_0. Our interpretation of the says connective allows for a very intuitive representation of the following form of delegation, which appears repeatedly in our case study: Principal k asserts that it will support the formula s if principal k' supports the formula s', but principal k has no authority over s'. This delegation can be represented in BL_0 by the formula: k says $((k' \text{ says } s') \supset s)$. Concretely, k only needs to sign the formula "$(k' \text{ says } s') \supset s$" with its private key to cause this delegation. If k' supports s', i.e., k' says s', then we can easily derive k says s using the axioms (I), (K) and the basic rule of modus ponens (if $\vdash r \supset r'$ and $\vdash r$, then $\vdash r'$):

1. By (I) on k' says s', derive k says k' says s'.
2. By (K) on k says $((k' \text{ says } s') \supset s)$ and modus ponens, derive $(k \text{ says } k' \text{ says } s') \supset (k \text{ says } s)$.
3. By modus ponens on (2) and (1), derive k says s.

(A sequent calculus proof of the same fact is shown in Appendix B.)

Neither of the other two interpretations of says mentioned at the beginning of this section allow for such an intuitive representation of this form of delegation. If k says $\cdot$ is interpreted as a normal necessitation modality as in [2, 24], then k says s cannot be derived from k says $((k' \text{ says } s') \supset s)$ and k' says s' because this interpretation of says does not admit the axiom (I). The "correct" representation of delegation in this logic is $(k' \text{ says } s') \supset (k \text{ says } s)$, not k says $((k' \text{ says } s') \supset s)$. However, unlike the latter formula, the former formula does not have a top-level says connective, so it not clear which principal's signed certificate will establish the delegation.

If, instead, k says $\cdot$ is interpreted as a lax modality as in [1, 18], then we face a different problem: In lax logic, from the delegation formula k says $((k' \text{ says } s') \supset s)$ and the formula k says s' (*not* the formula $(k' \text{ says } s')$), it is possible to derive k says s. Thus principal k retains control over the formula s', even as it delegates the formula to k', which may be undesirable, as in the following example from our case study:

```
admin claims (((indi/has-background K topsecret) :-
                    BA says (indi/has-ssbi K T),
                    T' = (T + 5y)) @ [T, T']) ∘ [−∞, +∞].
```

In words, the principal admin says that principal K has topsecret clearance if a person certified to check others' background, *BA*, says that K has passed a SSBI

(single scope background investigation). Although the predicate (indi/has-ssbi *K T*) has been delegated to *BA* by admin, the latter has no authority over this predicate. Consequently, this rule cannot be represented easily if says were to be interpreted as a lax modality. (Other, similar examples of delegation appear in Section 6.)

In summary, our interpretation of the says connective allows for a very intuitive and accurate representation of a specific and common form of delegation, which is not the case in earlier authorization logics.

6. Case Study Revisited

We revisit the case study of U.S. policies for access to sensitive intelligence information that was introduced in Section 2. In that section, we listed the top-level rules for access to a sensitive intelligence file and mentioned that authorization to read a classified file depends on the accessing individual possessing required clearances, represented by the predicate (indi/has-clearances/file *K F*) in the second rule of Section 2. In this section, we list the rules that define this predicate.

Verification of the predicate (indi/has-clearances/file *K F*) requires matching attributes of the file *F* to attributes of the individual *K*. Attributes of file *F* include its secrecy level (confidential, secret, or topsecret), citizenship requirements for access, and compartments (projects or divisions) that *F* is associated with. To access the file, *K* must have a secrecy clearance higher than that of the file, satisfy the file's citizenship requirements and be a member of all compartments the file is associated with. In the following we describe these attributes and discuss the policy rules that ascribe them to files and individuals. We start with a description of file attributes (Section 6.1), then describe attributes of principals (Section 6.2), and finally define the predicate (indi/has-clearances/file *K F*) (Section 6.3).

Notational conventions. We omit the suffix $\circ [-\infty, +\infty]$ from BL's judgments, abbreviating k claims $s \circ [-\infty, +\infty]$ to k claims s. As in Section 2, any variables in uppercase letters (*K*, *F*, etc.) in s are implicitly universally quantified immediately inside the prefix (k claims $\cdot$). (s :- $s_1, \ldots, s_n$) denotes (($s_1 \wedge \ldots \wedge s_n$) $\supset s$). We follow a descriptive naming convention for predicates. A predicate name has the form entity/attribute/..., where entity determines the entity whose attribute the predicate describes and attribute is a description of the property the predicate defines. Examples of entities are file and indi (individuals or principals). "..." may be any other relevant qualifiers. Common among these qualifiers is h which denotes a helper predicate that is used in the definition of the predicate without the h. The principal admin is the ultimate authority on all matters related to access and delegates part of its authority to other principals using BL rules presented later.

6.1. File Classification

When a file is classified, credentials are issued to determine its classification attributes, namely, its secrecy level, citizenship requirements for access, and its compartments. Each of these attributes must be established before the classified file can be accessed.

- Secrecy level: The secrecy level of a file is an indicator of the sensitivity of the contents of the file. It is one of confidential, secret, or topsecret, in increasing order of sensitivity.[4] Read access to a classified file is restricted to individuals who have secrecy clearance at a level equal to or greater than the secrecy level of the file.
- Citizenship requirement: A list of countries is associated with every classified file. Access is restricted only to citizens of those countries, and to those of the U.S. A commonly used abbreviation is "NOFORN" (no access to foreigners), which corresponds to an empty list of countries.
- Associated compartments: A compartment is a description of the purpose of a file, e.g., a project name or a division within the intelligence community. Every classified file is associated with zero or more compartments. Read access to a classified file is restricted only to those individuals who are associated with all compartments that the file is associated with (and possibly other compartments also).

The authority to decide which files need to be classified, and what secrecy level, citizenship requirements, and associated compartments apply to a classified file rests with very high ranking officers of the executive branch of the government and their representatives. These individuals are called Original Classification Authorities (OCAs). In our formal model, the predicate `indi/is-oca` O means that principal O is an OCA.

Compartment attributes. A compartment is created by an OCA. The OCA also fixes several compartment attributes that determine when an individual may be cleared into the compartment. Of these attributes, we model three prominent ones: (1) The minimum secrecy level at the which the individual must be cleared, (2) The minimum level of background check the individual must pass, and (3) Whether or not the individual has to pass a polygraph test. Formally, we define the predicate `compartment/is` C L L' B to mean that C is a valid compartment (in practice, C is a unique string naming the compartment), clearance into which requires:

- A secrecy clearance at level L or higher.
- A background check equivalent to that needed for secrecy clearance at level L' or higher.
- A polygraph test if the Boolean B is yes. Alternatively, if B is no, then a polygraph test is not necessary to be cleared into C.

Secrecy clearances, background checks and polygraph tests are described in Section 6.2. The following rule delegates the authority to create compartments from admin to every OCA O.

```
admin claims ((compartment/is C L L' B) :-
                        indi/is-oca O,
                        O says (compartment/is C L L' B)).
```

When a compartment is created it is assigned a special security officer (SSO), who manages the compartment. Written guidelines that determine what can and what can-

---

[4]There is another secrecy level called sbu (sensitive but unclassified), or "for official use only". Files at this level are *not classified* – sbu is merely a directive to officials to be more careful than usual when handling such files. Therefore, we do not consider sbu in our formalization.

not be classified into the compartment are also associated with the compartment. These guidelines constitute the security classification guide (SCG) of the compartment. In our formal model we abstract away the details of a SCG, and treat it only as a symbolic constant. Let the predicate `compartment/has-sso` *C S* mean that principal *S* is compartment *C*'s special security officer, and let (`compartment/has-scg` *C SCG*) mean that *SCG* is the security classification guide of compartment *C*.

In the following, we discuss BL rules for determining the secrecy level, citizenship requirements, and compartments associated with a file. The compartments associated with a file must be decided first since they are necessary to authorize the file's secrecy level and its citizenship requirements.

Determining a file's associated compartments. Let the predicate `file/has-compartments` *F CL* mean that file *F* is associated with exactly the compartments in the list *CL*. According to official guidelines, establishing this predicate requires two kinds of approvals: (a) An approval from an OCA stating that this should be the case, and (b) Approvals from the SSOs of all compartments in the list *CL* stating that the file may be associated with all the compartments in *CL*. Modeling the second requirement in BL is slightly tricky; we use a recursively defined helper predicate `file/has-compartments/h` *F CL CL′* which means that the SSOs of all compartments in *CL′* agree that *F* should be associated with all compartments in *CL*. The following rule uses this predicate with *CL′* = *CL* to allow a file to be associated with a list of compartments *CL*.

admin claims ((`file/has-compartments` *F CL*) :-
                                         `indi/is-oca` *O*,
                                         O says (`file/has-compartments` *F CL*),
                                         `file/has-compartments/h` *F CL CL*).

The following two rules define the helper predicate (`file/has-compartments/h` *F CL CL′*) by induction on *CL′*. The symbol nil denotes the empty list and | is an infix binary function that concatenates an element to a list.

admin claims (`file/has-compartments/h` *F CL* nil).

admin claims ((`file/has-compartments/h` *F CL* (*C′* | *CL′*)) :-
                                        `compartment/has-sso` *C′ S*,
                                        S says (`file/has-compartments` *F CL*),
                                        `file/has-compartments/h` *F CL CL′*).

The second rule above means that admin will trust that the SSOs of all compartments in *C′* | *CL′* agree that *F* should be associated with the compartments in *CL* if (a) The SSO *S* of compartment *C′* agrees to this fact (first two conditions of the rule) and (b) Recursively, the SSOs of all compartments in *CL′* agree to this fact (third condition).

Determining a file's secrecy level. According to official guidelines, a file's secrecy level may be set to *L* if: (a) An OCA says that this should be case, and (b) The SSOs of all compartments associated with the file agree that the SCGs of their respective compartments allow the file to be given secrecy level *L*. Formally, let the predicate `file/has-level` *F L* mean that file *F* has secrecy level *L*, and `file/has-level/h` *F L CL* mean that the SSOs of all compartments in the list *CL* agree that *F* may be given secrecy level *L* in accordance with their respective SCGs. Then, the

following rule formalizes the above conditions for assigning the secrecy level *L* to file *F*.

```
admin claims ((file/has-level F L) :- indi/is-oca O,
                                       O says (file/has-level F L),
                                       file/has-compartments F CL,
                                       file/has-level/h F L CL).
```

The following two rules define the predicate `file/has-level/h` *F L CL* by induction on the list *CL*. The predicate `file/has-level/scg` *F L SCG* is intended to mean that the security classification guide *SCG* mandates that file *F* be given secrecy level *L*.

```
admin claims (file/has-level/h F L nil).
```

```
admin claims ((file/has-level/h F L (C′ | CL′)) :-
                                       compartment/has-sso C′ S,
                                       compartment/has-scg C′ SCG,
                                       S says (file/has-level/scg F L SCG),
                                       file/has-level/h F L CL′).
```

According to the second rule above, `admin` asserts that the SSOs of all compartments in *C′ | CL′* agree that *F* should have secrecy level *L* if (a) The SSO *S* of *C′* states that this assignment of level would be in accordance with the SCG of *C′* (third condition of the rule), and (b) Recursively, the SSOs of all compartments in *CL′* agree with this assignment (fourth condition). It follows from these rules that if no compartments are associated with a file *F*, i.e., if `admin says (file/has-compartments` *F* `nil)`, then an OCA *O*'s statement *O* `says (file/has-level` *F L*`)` suffices to give a security level *L* to *F*.

The second rule above is another example of the kind of delegation mentioned in Section 5.4 because the rule transfers authority over the predicate `file/has-level/scg` from principal `admin` to principal *S*, but `admin` has no jurisdiction over the predicate.

Determining a file's citizenship requirements.   Determining the citizenship requirements for reading a file is similar to determining the file's secrecy level, so we omit its details. Briefly, an OCA approves the list of countries to whose citizens access must be restricted, and the SSOs of all compartments associated with the file must certify that this list would be allowed by their respective SCGs. The predicate `file/has-citizenship` *F UL* means that reading file *F* requires a citizenship of one of the countries in the list *UL* (or of the U.S.).

Summary of file classification.   In summary, classifying a file requires credentials to determine associated compartments, its secrecy level and citizenship requirements for access to it. These credentials are issued by various individuals including an OCA and the SSO's of all compartments the file is associated with. In practice, any issued credential will be valid for only a stipulated duration of time. For example, if an OCA *O* says that file *F* should have secrecy level *L* from 2009 to 2011, this would be represented in BL as (*O* claims (`file/has-level` *F L*)) ∘ [2009:01:01:00:00:00, 2011:12:31:23:59:59]. BL's inference rules propagate these time restrictions to other facts derived from the credentials and policy rules.

6.2. Individual Clearances

Individuals require clearance both at secrecy levels and into compartments, as well as citizenship of specific countries to read classified files. We call these three primary clearances of individuals. In order to obtain primary clearances, other auxiliary clearances are needed. These include polygraph tests and background checks. In this section we formalize the rules for obtaining auxiliary clearances, as well as rules for combining them to determine primary clearances. We start with the auxiliary clearances.

6.2.1. Auxiliary Clearances

Polygraph clearance. Individuals may have to pass a polygraph test to get clearance into certain compartments. Polygraph tests are administered and certified by trained individuals, whom we call polygraph administrators. The procedures for identifying polygraph administrators are beyond the scope of our formalization; we simply assume that the predicate `indi/is-polygraph-admin` *PA* means that principal *PA* is a trusted polygraph administrator. Let `indi/has-polygraph` *K* mean that principal *K* has passed a polygraph test. The following rule states that if *PA* is a polygraph administrator, and *PA* says that *K* has passed a polygraph test, then admin will trust the latter.

```
admin claims ((indi/has-polygraph K) :-
                           indi/is-polygraph-admin PA,
                           PA says (indi/has-polygraph K)).
```

Background checks. A background check is necessary to get clearance both at secrecy levels and into compartments. There are two commonly used background checks: (1) National Agency Check with Local Agency Check and Credit Check or NACLC, and (2) Single Scope Background Investigation or SSBI. NACLC is an investigation of an individual's criminal records and credit history. SSBI includes the NACLC and in addition requires interviews of colleagues and investigation of family history. We assume that certain principals called background administrators are certified to check others' backgrounds.

From the perspective of formalization, it is very convenient to abstract background checks by the secrecy level for which they are mandatory. For example, a *background check at level* confidential would correspond to a background check that is needed to get clearance at secrecy level confidential. This abstraction is useful because official guidelines mandate that background checks expire after fixed intervals of time that depend on the secrecy level for which the checks are conducted. The actual check corresponding to each secrecy level and its expiration time is shown in the table below.

Abstract level of background check	Actual background check needed and expiration
confidential	NACLC, expires in 15 years
secret	NACLC, expires in 10 years
topsecret	SSBI, expires in 5 years

Let `indi/is-background-admin` *BA* mean that principal *BA* is a background administrator, i.e., certified to conduct background checks. Further, let `indi/has-naclc` *K T* mean that principal *K* passed an NACLC at time *T*, and `indi/has-ssbi` *K T* mean

that principal K passed an SSBI at time T. The following rules define the predicate `indi/has-background` $K\,L$, which means that principal K has a background check that is needed for clearance at secrecy level L. There are three rules, one for each possible value of L. A salient point to observe is the use of the @ connective for automatically expiring background checks in accordance with the table above. The symbol y following a number means "years". Hence, $15y$ means 15 years. As an example, the first rule below means that if BA is a background administrator and BA states that K passed an NACLC at time T, then admin asserts that K has a background check at level confidential in the interval $[T, T + 15y]$.

admin claims $(((\texttt{indi/has-background}\ K\ \text{confidential})$:-
$\qquad\qquad\qquad\qquad\qquad \texttt{indi/is-background-admin}\ BA,$
$\qquad\qquad\qquad\qquad\qquad BA\ \textsf{says}\ (\texttt{indi/has-naclc}\ K\ T))\ @\ [T, T + 15y]).$

admin claims $(((\texttt{indi/has-background}\ K\ \text{secret})$:-
$\qquad\qquad\qquad\qquad\qquad \texttt{indi/is-background-admin}\ BA,$
$\qquad\qquad\qquad\qquad\qquad BA\ \textsf{says}\ (\texttt{indi/has-naclc}\ K\ T))\ @\ [T, T + 10y]).$

admin claims $(((\texttt{indi/has-background}\ K\ \text{topsecret})$:-
$\qquad\qquad\qquad\qquad\qquad \texttt{indi/is-background-admin}\ BA,$
$\qquad\qquad\qquad\qquad\qquad BA\ \textsf{says}\ (\texttt{indi/has-ssbi}\ K\ T))\ @\ [T, T + 5y]).$

The remaining policy rules refer only to the predicate `indi/has-background` $K\,L$, not to the predicates `indi/has-naclc` $K\,T$ and `indi/has-ssbi` $K\,T$.

6.2.2. Primary Clearances

An individual's clearance at a secrecy level, clearance into compartments, as well as citizenship directly determine what classified files she has access to. We now describe rules that define how clearances are determined.

Clearance at secrecy levels.   As mentioned earlier, an individual must pass a background check at level L in order to get clearance at secrecy level L. In addition, the individual must have a *need to get the clearance*. Since the factors determining this need are varied and are not completely specified, we simply assume that the predicate `indi/needs-level` $K\,L$ means that principal K has a need to get clearance at secrecy level L. Let `indi/has-level` $K\,L$ mean that individual K has clearance at secrecy level L. `level/below` $L\,L'$ means that level L is below the level L' in the order confidential < secret < topsecret. It is defined later. The following rule states that K has clearance at secrecy level L if K needs this clearance, and K has passed a background check at some level L' which is higher than L.

admin claims $((\texttt{indi/has-level}\ K\ L)$:-
$\qquad\qquad\qquad\qquad\ \texttt{indi/needs-level}\ K\ L,$
$\qquad\qquad\qquad\qquad\ \texttt{indi/has-background}\ K\ L',$
$\qquad\qquad\qquad\qquad\ \texttt{level/below}\ L\ L').$

As explained earlier, the validity of `indi/has-background` $K\,L'$ is limited to 15, 10, or 5 years depending on L'. The above rule and the inference rules of BL transfer the same restrictions to `indi/has-level` $K\,L$. The predicate `level/below` is defined by the following rules.

admin claims (level/below $L\,L$).

admin claims (level/below confidential secret).

admin claims (level/below secret topsecret).

admin claims (level/below confidential topsecret).

Clearance into compartments. To be cleared into a compartment, an individual must satisfy all its requirements – secrecy level, background check, and a polygraph test if needed. These requirements are uniquely determined from the predicate compartment/is $C\,L\,L'\,B$, which is established when the compartment C is created (as explained in Section 6.1). Let the predicates indi/has-comp-level $K\,C$, indi/has-comp-background $K\,C$, and indi/has-comp-polygraph $K\,C$ mean that an individual has clearance at an appropriate secrecy level, background check, and polygraph check (if needed) for being cleared into compartment C. The following rules define these predicates by considering respectively the 2nd, 3rd, and 4th arguments of the predicate compartment/is $C\,L\,L'\,B$. An underscore _ represents an implicitly named variable, whose instantiated value is irrelevant to the rule.

admin claims ((indi/has-comp-level $K\,C$) :-
$\qquad\qquad\qquad\qquad$ compartment/is $C\,L$ _ _,
$\qquad\qquad\qquad\qquad$ indi/has-level $K\,L''$,
$\qquad\qquad\qquad\qquad$ level/below $L\,L''$).
admin claims ((indi/has-comp-background $K\,C$) :-
$\qquad\qquad\qquad\qquad$ compartment/is C _ L' _,
$\qquad\qquad\qquad\qquad$ indi/has-background $K\,L''$,
$\qquad\qquad\qquad\qquad$ level/below $L'\,L''$).
admin claims ((indi/has-comp-polygraph $K\,C$) :-
$\qquad\qquad\qquad\qquad$ compartment/is C _ _ yes,
$\qquad\qquad\qquad\qquad$ indi/has-polygraph K).
admin claims ((indi/has-comp-polygraph $K\,C$) :-
$\qquad\qquad\qquad\qquad$ compartment/is C _ _ no).

Using the above predicates, we define the predicate indi/has-compartment $K\,C$ which means that an individual K is cleared into the compartment C. An important fact to observe here is that in addition to satisfying the three requirements of the compartment, the SSO S of the compartment must certify the clearance, and, as in the case of clearance at secrecy levels, the principal must actually need the clearance (predicate indi/needs-compartment $K\,C$).

admin claims ((indi/has-compartment $K\,C$) :-
$\qquad\qquad\qquad\qquad$ indi/needs-compartment $K\,C$
$\qquad\qquad\qquad\qquad$ compartment/has-sso $C\,S$,
$\qquad\qquad\qquad\qquad$ S says (indi/has-compartment $K\,C$),
$\qquad\qquad\qquad\qquad$ indi/has-comp-level $K\,C$,
$\qquad\qquad\qquad\qquad$ indi/has-comp-background $K\,C$,
$\qquad\qquad\qquad\qquad$ indi/has-comp-polygraph $K\,C$).

Finally, the following two rules define a related, useful predicate indi/has-compartment/list $K\,CL$ which means that K is cleared into *all compart-*

ments in the list *CL*.

```
admin claims (indi/has-compartment/list K nil).
```

```
admin claims ((indi/has-compartment/list K (C | CL)) :-
                                    indi/has-compartment K C,
                                    indi/has-compartment/list K CL).
```

6.3. Clearances to Classified Files

Building on predicates defined in the previous sections, we now provide rules that define the central predicate (indi/has-clearances/file K F). First, we define the following three auxiliary predicates using straightforward rules shown below: (a) indi/has-level/file K F, which means that principal K has secrecy clearance at a level higher than that of file F, (b) indi/has-comps/file K F, which means that principal K is cleared into all compartments that F is associated with, and (c) indi/has-cit/file K F, which means that principal K is a citizen of at least one country in the citizenship requirement list of F.

```
admin claims ((indi/has-level/file K F) :-
                            file/has-level F L,
                            indi/has-level K L',
                            level/below L L').
admin claims ((indi/has-comps/file K F) :-
                            file/has-compartments F CL,
                            indi/has-compartment/list K CL).
admin claims ((indi/has-cit/file K F) :-
                            file/has-citizenship F UL,
                            indi/has-citizenship/list K UL).
admin claims ((indi/has-cit/file K F) :-
                            indi/has-citizenship K usa).
```

The predicate indi/has-citizenship/list K UL means that K is a citizen of at least *one of the countries* in the list *UL*. The last rule above means that any U.S. citizen satisfies the citizenship requirement for reading a file, irrespective of the latter's actual citizenship requirements. The following rule defines the predicate indi/has-clearances/file K F.

```
admin claims ((indi/has-clearances/file K F) :-
                                    indi/has-level/file K F,
                                    indi/has-comps/file K F,
                                    indi/has-cit/file K F).
```

7. Related Work

Several formal frameworks for authorization policies allow for representation of state, but no prior proposal has considered an integration of state and logic from a proof-theoretic perspective. Perhaps closest to BL's treatment of stateful atoms is the Nexus Authorization Logic (NAL) [34] that is used for authorizing access in several compo-

nents of the Nexus operating system. NAL includes support for state predicates in a manner similar to that stipulated in Section 3.2, i.e., the reference monitor verifies certain predicates using trusted decision procedures that may refer to the system state. Several other logic-based frameworks for representing authorization policies [5, 8, 10, 25] do not make a distinction between constraints and state predicates, and consequently support system state implicitly as part of their support for constraints. However, we believe that maintaining this distinction is important from the perspective of both implementation and reasoning about policies expressed in logic.

There has also been some work on declarative languages and logics in which authorization policies and *state transitions* can be represented and reasoned about simultaneously [7, 9, 15]. In contrast, BL's state predicates are meant to model situations where rules for state transitions are not specified. Some recent programming languages, e.g., [11, 12], use type systems to enforce state-dependent authorization policies that are represented in first-order logic. Stateful atoms are not distinguished from others in the proof theory used in these languages.

The connective k says s has been included in several past proposals for writing access policies, starting with the work of Abadi et al [2]. The BL connective $s @ [u_1, u_2]$ is based on our prior work with DeYoung [14], and our treatment of constraints goes further back to work on reconciling constraint domains and proof theory of linear logic [22, 33]. Study of proof theory for authorization logics was initiated in our prior work [18]. The present paper incorporates many ideas from that work, especially the use of intuitionistic first-order logic as a foundation for authorization policies.

8. Conclusion

A proof-theoretic treatment of state in an authorization logic requires careful design. Part of the complication arises due to the well-understood difficulty of reconciling external verification procedures with proof theory, but most of the design choices arise in the interaction between state predicates and other features of authorization logic, in particular, explicit time. The logic BL strikes a good balance in this design space, as evident from its strong metatheoretic foundations and validation through a realistic case study of policies for access to sensitive intelligence information in the U.S. It makes several important design decision, e.g., it does not treat time as a special case of state to improve expressiveness and limits the interaction between state and time to avoid representation of unenforceable policies.

Several avenues remain for future work. We have not yet explored semantic models of BL, e.g., standard Kripke models. We expect this to be interesting for BL because its judgments include time intervals via the suffix $\circ [u_1, u_2]$ so it may be possible to prove soundness and completeness over Kripke models whose possible worlds are intervals of time. This would provide an independent justification to some of our design choices with respect to time. The interpretation of state would likely be non-obvious to remain consistent with our decision to treat $i \circ [u_1, u_2]$ independent of the interval $[u_1, u_2]$ for any stateful atom i.

A second, related avenue of future work is decidability problems for fragments of BL. The entire logic BL is clearly undecidable because it is a superset of first-order logic. However, because decision procedures could be practically useful in implementing the

logic, it would be useful to find decidable fragments of the logic that are reasonably expressive.

References

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

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

[3] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 52–62, 1999.

[4] Kumar Avijit, Anupam Datta, and Robert Harper. Distributed programming with distributed authorization. In *Proceedings of the Fifth ACM Workshop on Types in Language Design and Implementation (TLDI)*, 2009.

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

[6] Lujo Bauer, Scott Garriss, Jonathan M. McCune, Michael K. Reiter, Jason Rouse, and Peter Rutenbar. Device-enabled authorization in the Grey system. In *8th Information Security Conference (ISC)*, pages 431–445, 2005.

[7] Moritz Y. Becker. Specification and analysis of dynamic authorisation policies. In *22nd IEEE Computer Security Foundations Symposium (CSF)*, pages 203–217, 2009.

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

[9] Moritz Y. Becker and Sebastian Nanz. A logic for state-modifying authorization policies. In *12th European Symposium on Research in Computer Security (ESORICS)*, pages 203–218, 2008.

[10] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management applied to health records. In *17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 139–154, 2004.

[11] Johannes Borgström, Andrew D. Gordon, and Riccardo Pucella. Roles, stacks, histories: A triple for Hoare. Technical Report MSR-TR-2009-97, Microsoft Research, 2009.

[12] Niklas Broberg and David Sands. Paralocks: Role-based information flow control and beyond. *SIGPLAN Notices*, 45(1):431–444, 2010.

[13] John DeTreville. Binder, a logic-based security language. In *Proceedings of the IEEE 2002 Symposium on Security and Privacy (S&P'02)*, pages 105–113, May 2002.

[14] Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. In *21st IEEE Computer Security Foundations Symposium (CSF)*, pages 133–145, 2008. Extended version available as Carnegie Mellon University Technical Report CMU-CS-07-166.

[15] Henry DeYoung and Frank Pfenning. Reasoning about the consequences of authorization policies in a linear epistemic logic, 2009. Workshop on Foundations of Computer Security (FCS), `http://www.cs.cmu.edu/~hdeyoung/papers/fcs09.pdf`.

[16] M. Fairtlough and M.V. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, August 1997.

[17] Deepak Garg. *Proof Theory for Authorization Logic and Its Application to a Practical File System*. PhD thesis, Carnegie Mellon University, 2009. Available as Technical Report CMU-CS-09-168. Online at `http://www.mpi-sws.org/~dg/papers.html`.

[18] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In *19th Computer Security Foundations Workshop (CSFW)*, pages 283–293, 2006.

[19] Deepak Garg and Frank Pfenning. A proof-carrying file system. In *Proceedings of the 31st IEEE Symposium on Security and Privacy (Oakland)*, pages 349–364, 2010.

[20] Deepak Garg, Frank Pfenning, Denis Serenyi, and Brian Witten. A logical representation of common rules for controlling access to classified information. Technical Report CMU-CS-09-139, Carnegie Mellon University, 2009. Available online at `http://www.mpi-sws.org/~dg/papers.html`.

[21] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

[22] Limin Jia. *Linear Logic and Imperative Programming*. PhD thesis, Department of Computer Science, Princeton University, 2008.

[23] Limin Jia, Jeffrey A. Vaughan, Karl Mazurak, Jianzhou Zhao, Luke Zarko, Joseph Schorr, and Steve Zdancewic. Aura: A programming language for authorization and audit. In *Proceedings of the International Conference on Functional Programming (ICFP)*, pages 27–38, 2008.

[24] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, 1992.

[25] Ninghui Li, John C. Mitchell, and W.H. Winsborough. Design of a role-based trust-management framework. In *23rd IEEE Symposium on Security and Privacy (Oakland)*, pages 114–130, 2002.

[26] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.

[27] Office of the Director of Central Intelligence. DCID 1/19: Security policy for sensitive compartmented information and security policy manual, 1995. `http://www.fas.org/irp/offdocs/dcid1-19.html`.

[28] Office of the Director of Central Intelligence. DCID 1/7: Security controls on the dissemination of intelligence information, 1998. `http://www.fas.org/irp/offdocs/dcid1-7.html`.

[29] Office of the Press Secretary of the White House. Executive order 12958: Classified national security information, 1995. `http://www.fas.org/sgp/clinton/eo12958.html`.

[30] Office of the Press Secretary of the White House. Executive order 13292: Further amendment to executive order 12958, as amended, classified national security information, 2003. `http://nodis3.gsfc.nasa.gov/displayEO.cfm?id=EO_13292_`.

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

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

[33] Uluç Saranli and Frank Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems. In *International Conference on Robotics and Automation (ICRA)*, pages 3705–3710, 2007.

[34] Fred B. Schneider, Kevin Walsh, and Emin Gün Sirer. Nexus Authorization Logic (NAL): Design rationale and applications. *ACM Transactions on Information and System Security*, 14(1):8:1–8:28, 2011.

A. Inference Rules of BL

This appendix lists all the inference rules of BL's sequent calculus (Section 3.3.1).

$$\frac{\Sigma;\Psi \models u_1' \leq u_1 \qquad \Sigma;\Psi \models u_2 \leq u_2'}{\Sigma;\Psi;E;\Gamma, p \circ [u_1',u_2'] \xrightarrow{v} p \circ [u_1,u_2]}\text{init}$$

$$\frac{\Sigma;\Psi;E;\Gamma, k \text{ claims } s \circ [u_1,u_2], s \circ [u_1,u_2] \xrightarrow{v} r \circ [u_1',u_2']}{v = k, u_b, u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2}{\Sigma;\Psi;E;\Gamma, k \text{ claims } s \circ [u_1,u_2] \xrightarrow{v} r \circ [u_1',u_2']}\text{claims}$$

$$\frac{\Sigma;\Psi;E;\Gamma \xrightarrow{v} s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (s @ [u_1,u_2]) \circ [u_1',u_2']}@\text{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma, s \circ [u_1,u_2] \xrightarrow{v} r \circ [u_1'',u_2'']}{\Sigma;\Psi;E;\Gamma, (s @ [u_1,u_2]) \circ [u_1',u_2'] \xrightarrow{v} r \circ [u_1'',u_2'']}@\text{L}$$

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} i \circ [u_1,u_2]}\text{stateR} \qquad \frac{\Sigma;\Psi;E,i;\Gamma \xrightarrow{v} r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Gamma, i \circ [u_1,u_2] \xrightarrow{v} r \circ [u_1',u_2']}\text{stateL}$$

$$\frac{\Sigma;\Psi \models c}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} c \circ [u_1,u_2]}\text{consR} \qquad \frac{\Sigma;\Psi,c;E;\Gamma \xrightarrow{v} r \circ [u'_1,u'_2]}{\Sigma;\Psi;E;\Gamma,c \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}\text{consL}$$

$$\frac{\Sigma;\Psi;E;\Gamma\downarrow \xrightarrow{k,u_1,u_2} s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (k \text{ says } s) \circ [u_1,u_2]}\text{saysR}$$

$$\frac{\Sigma;\Psi;E;\Gamma,k \text{ claims } s \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}{\Sigma;\Psi;E;\Gamma,(k \text{ says } s) \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}\text{saysL}$$

$$\frac{\Sigma,x_1\text{:time},x_2\text{:time};\Psi,u_1 \leq x_1,x_2 \leq u_2;E;\Gamma,s_1 \circ [x_1,x_2] \xrightarrow{v} s_2 \circ [x_1,x_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (s_1 \supset s_2) \circ [u_1,u_2]}\supset\text{R}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Gamma,(s_1 \supset s_2) \circ [u_1,u_2] \xrightarrow{v} s_1 \circ [u'_1,u'_2] \\ \Sigma;\Psi;E;\Gamma,(s_1 \supset s_2) \circ [u_1,u_2],s_2 \circ [u'_1,u'_2] \xrightarrow{v} r \circ [u''_1,u''_2] \\ \Sigma;\Psi \models u_1 \leq u'_1 \qquad \Sigma;\Psi \models u'_2 \leq u_2\end{array}}{\Sigma;\Psi;E;\Gamma,(s_1 \supset s_2) \circ [u_1,u_2] \xrightarrow{v} r \circ [u''_1,u''_2]}\supset\text{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma \xrightarrow{v} s_1 \circ [u_1,u_2] \qquad \Sigma;\Psi;E;\Gamma \xrightarrow{v} s_2 \circ [u_1,u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (s_1 \wedge s_2) \circ [u_1,u_2]}\wedge\text{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma,s_1 \circ [u_1,u_2],s_2 \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}{\Sigma;\Psi;E;\Gamma,(s_1 \wedge s_2) \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}\wedge\text{L} \qquad \frac{}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} \top \circ [u_1,u_2]}\top\text{R}$$

$$\frac{}{\Sigma;\Psi;E;\Gamma,\bot \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}\bot\text{L} \qquad \frac{\Sigma,x\text{:}\sigma;\Psi;E;\Gamma \xrightarrow{v} s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{v} (\forall x\text{:}\sigma.s) \circ [u_1,u_2]}\forall\text{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma,(\forall x\text{:}\sigma.s) \circ [u_1,u_2],s[t/x] \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Psi;E;\Gamma,(\forall x\text{:}\sigma.s) \circ [u_1,u_2] \xrightarrow{v} r \circ [u'_1,u'_2]}\forall\text{L}$$

B. Sequent Calculus Proof from Section 5.4

This appendix lists a BL_0 sequent calculus proof of k says s from the hypotheses k says $((k' \text{ says } s') \supset s)$ and k' says s'. We simplify notation slightly by abbreviating the judgment s true to s. α is an arbitrary principal. Proof branches labeled (Identity) are applications of the identity theorem for BL_0 (analogue of Theorem 4.4 of BL).

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{\text{(Identity)}}{\Sigma; s' \xrightarrow{k'} s'}
      }{\Sigma; k' \text{ claims } s' \xrightarrow{k'} s'} \text{ claims}
    }{\Sigma; k' \text{ claims } s' \xrightarrow{k} k' \text{ says } s'} \text{ saysR}
    \qquad
    \dfrac{\text{(Identity)}}{\Sigma; s \xrightarrow{k} s}
  }{
    \dfrac{
      \dfrac{
        \Sigma; (k' \text{ says } s') \supset s, k' \text{ claims } s' \xrightarrow{k} s
      }{\Sigma; k \text{ claims } ((k' \text{ says } s') \supset s), k' \text{ claims } s' \xrightarrow{k} s} \text{ claims}
    }{\Sigma; k \text{ claims } ((k' \text{ says } s') \supset s), k' \text{ claims } s' \xrightarrow{\alpha} k \text{ says } s} \text{ saysR}
  } \supset\!\text{L}
}{\Sigma; k \text{ says } ((k' \text{ says } s') \supset s), k' \text{ says } s' \xrightarrow{\alpha} k \text{ says } s} \text{ saysL} \times 2
$$