

An Authorization Logic with Explicit Time*

Henry DeYoung

Deepak Garg

Frank Pfenning

Carnegie Mellon University

E-mail: hdeyoung@andrew.cmu.edu, {dg, fp}@cs.cmu.edu

Abstract

We present an authorization logic that permits reasoning with explicit time. Following a proof-theoretic approach, we study the meta-theory of the logic, including cut elimination. We also demonstrate formal connections to proof-carrying authorization’s existing approach for handling time and comment on the enforceability of our logic in the same framework. Finally, we illustrate the expressiveness of the logic through examples, including those with complex interactions between time, authorization, and mutable state.

1. Introduction

Most secure systems restrict operations that users, machines, and other principals can perform on files and other resources. A reference monitor authorizes (or denies) requests to access resources, in consultation with a set of rules called the security policy. Time is central to most policies. A student, for instance, may be allowed to access course related material *only* during the specific semester that she is registered for the class.

In practice, security policies are often large and complicated, necessitating formal mechanisms for both their enforcement and their analysis. Although several trust management frameworks [8, 9, 25, 29–31, 35], languages [7, 14], and access control logics [1, 2, 12, 19, 20, 27] have been proposed for enforcing and sometimes for reasoning about access control policies, these proposals rarely handle time explicitly, either omitting it altogether or leaving it to an external enforcement mechanism. As a result, policies with complex time-dependent relationships cannot be expressed, and, in other cases, reasoning accurately about time is extremely difficult.

The purpose of this paper is to bridge this gap between

using time in practice and reasoning about it; we propose an authorization logic that allows explicit mention of time, making it easier to reason about the time-dependent consequences of policies. This logic combines ideas from an existing authorization logic [19, 20] with ideas from both hybrid logics [11, 36] and constraint-based logics [24, 37] to allow propositions of the form $A @ I$ where A is a proposition and I is the time interval on which it holds. Following earlier proposals, we make the logic constructive to keep evidence as direct as possible. We also include linearity to model consumable authorizations; their use is illustrated in our examples. We call the logic η -logic (pronounced *eta logic* for *Explicitly Timed Authorization logic*).

η -logic is strictly more expressive than existing logics for access control because policies with complex time-dependent relationships can be expressed in it. For instance, the following policy can be expressed in η -logic: “If an employee requests a parking space before the end of a month, she will be given a parking permit valid throughout the next month.” It is difficult to imagine how such a policy could be expressed unless time is allowed explicitly in formulas.

Our principal interest in designing η -logic is its deployment with proof-carrying authorization (PCA) [3–5, 28]. In the PCA paradigm, security policies are formalized in a logic, and each access request is accompanied by a formal proof establishing that authorization for the request follows logically from the policies. The reference monitor verifies the correctness of the proof, and allows or denies access accordingly. PCA provides a flexible mechanism for access control in distributed systems. In existing approaches, the proof presented to the reference monitor establishes that the requester is allowed to access the resource in question, leaving the validity of the proof at the *time* of request to a separate enforcement check. With η -logic, the proof itself can be refined to mention that access is allowed at the time of the request. We make a formal connection between the two approaches in Section 3.4.

In addition to its applications with PCA, we expect that η -logic can be used in specifying the behavior of systems with time-dependent authorization policies. In such cases, the logic can be used to formally establish correctness prop-

*This work has been partially supported by the Air Force Research Laboratory grant FA87500720028 *Accountable Information Flow via Explicit Formal Proof* and the National Science Foundation grant NSF-0716469 *Manifest Security*.

erties of the system, as in [19]. The first example in Section 4 illustrates this approach. Linearity plays a crucial role in this setting, facilitating accurate models of mutable state.

In the spirit of Gentzen’s pioneering work on proof theory [21], we abstain from model-theoretic semantics, instead presenting η -logic as a sequent calculus. This brings the logic closer to realization in tools like theorem provers and proof verifiers, and facilitates a detailed study of its meta-theory, which is central to our work. We establish several properties, including consistency and cut elimination, which increase confidence in the logic’s foundations. Cut elimination also implies that the meanings of connectives in the logic are independent of each other. This makes the logic open to extension with new connectives. Establishing these properties is non-trivial, involving a deep interplay between inference rules and constraints.

In summary, this work makes several contributions. First, it introduces explicit time into reasoning about authorization. In contrast to existing approaches, this makes it possible to express and reason robustly about time-dependent policies.

Second, it formalizes implementations of PCA that deal with time in an extra-logical manner and rely only on validity intervals of embedded digitally signed certificates. We further show that policy enforcement in PCA (at least for a fragment) is no more difficult than in the logics that have been proposed previously.

Third, our system integrates explicit time and linearity. This represents a non-trivial challenge because both time intervals and single-use assumptions restrict the availability of hypotheses during reasoning, but in entirely different ways. Our meta-theorems, specifically the cut elimination and identity properties, show that these concepts are indeed compatible, at least in a constructive setting. The key is a novel combination of ideas from hybrid logic with constraints. The examples demonstrate that the combination allows logical expression and enforcement of a wide range of naturally occurring policies which were previously intractable.

Related work. Our work draws upon ideas from several kinds of logics. Most closely related are works on constructive authorization logic [19, 20], from which we borrow linearity, affirmation, and our style of presentation. The “says” construct in our logic was first introduced by Abadi *et al.* [2, 27], and adopted by almost all subsequent proposals.

The formalization of time in our presentation combines ideas from both hybrid logics [11, 36] and constraint-based logics [24, 37]. Such a combination has been studied to a limited extent in Temporal Annotated Constraint Logic Programming (TACLP) [18]. This work, done in the context of

logic programming without authorizations, allows interval annotations on atomic formulas, similar to our $A @ I$ construct. Besides TACLP, we are unaware of any work that uses hybrid logic for modeling time.

Linearity, which is important for modeling consumable resources, was introduced in a logic by Girard [22]. The judgmental form of linear logic was first studied by Chang *et al.* [13]. The use of linearity in conjunction with authorization was first proposed by two of the present authors and others [19]. Some enforcement mechanisms in the distributed setting have also been described [10].

More broadly, this work relates to languages and logics for expressing and enforcing access control policies [1, 2, 7–9, 12, 14, 19, 20, 25, 27–31, 35, 38]. With the exception of the policy language SecPAL [7], we are not aware of explicit use of time in any of these proposals. SecPAL’s enforcement of time is external to the language, based on a constraint system that is not reasoned about within the formal semantics. In contrast, our logic permits direct reasoning with validity of formulas. It would be interesting to study the potential formal connections between the two approaches.

The principal target of our design is proof-carrying authorization (PCA) [3–5, 28]. In existing work on PCA, validity of certificates plays an integral part, but it is not included in the logic. Rather, it is enforced by a separate check. In η -logic this check becomes part of proof-verification.

An alternate approach to reasoning about time is based on temporal logics [16]. There, one reasons about events *relative* in time to others. Although sometimes useful in reasoning about security protocols, the approach appears to be ineffective in the context of security policies and PCA, which rely heavily on absolute time.

Another related line of work is interval temporal logic [33], where one reasons about sequences of states in an evolving system. Like temporal logic, the method seems inadequate for reasoning about authorization policies.

Finally, Kanovich *et al.* [26] have studied encodings of real-time systems in linear logic using a distinguished predicate for the current time. However, this method also appears to lack the expressive power necessary for authorization policies; it does not seem to permit conjunctions where each conjunct is qualified with a different time, for example.

Organization of the paper. In Section 2, we introduce our logic and its proof system, and study its meta-theory. Section 3 describes η -logic’s application to PCA. Section 4 illustrates the expressiveness of the logic by showing examples that contain complex time-dependent relationships. Section 5 concludes the paper.

Due to lack of space, proofs of theorems presented in this

paper are omitted. They may be found in the companion technical report [15].

2. η -logic: Authorization with explicit time

At its core, η -logic is a first-order intuitionistic logic. It integrates several other constructs: affirmations, linearity, hybrid worlds representing time, and constraints. While these constructs have been studied separately in the past, their interaction with each other is deep and non-trivial. In particular, the hybrid nature of η -logic interacts with all the other components, making it impossible to construct η -logic as an extension of either linear logic or a logic of affirmation without changing the nature of the underlying judgments.

Following Per Martin-Löf [32], we use a judgmental approach in describing the logic: we separate formulas from judgments, making the latter the objects of reasoning. (See [34] for a detailed discussion of this approach in the context of modal logic.) In the interest of readability, we describe the logic in several steps. We begin by briefly describing the structure of first-order terms and sorts. Next, we describe the judgments that capture time, linearity, and affirmation. We then discuss constraints, and finally present the logic’s connectives and proof rules.

2.1. First-order terms and sorts

We assume that quantifiable terms can be typed into different sorts (denoted by the meta-variable s). We stipulate at least two sorts: a sort of principals (principal) and a sort of intervals of time (interval). If t is a term and Σ assigns sorts to all constants occurring in t , we write $\Sigma \vdash t:s$ to mean that the term t has sort s . We write $[t/x]A$ to denote the formula obtained by substituting the term t for all free occurrences of x in A .

Principals, denoted by the letter K , represent machines, users, or programs that make access requests or issue policies. Concretely, they may be simple bit strings that represent names, identifiers, or keys.

Intervals, denoted by the letter I , represent *sets of time points* over which formulas are true. Borrowing terminology from hybrid logic, they are worlds which qualify formulas. We do not fix structures for either time points or their sets, but postulate necessary conditions that must hold on them. These conditions are described in Section 2.3. Intuitively, one may think of time points as points on the real line, and sets I as closed intervals on the real line. However, it should be noted that the term interval is really a misnomer here: we could work with other kinds of sets as well. In many natural scenarios, the sets are intervals, and therefore we continue to use this nomenclature.

2.2. Judgments

Ordinarily, logic is concerned with the truth of formulas without reference to time. However, in access control, the truth of formulas changes with time. For instance, if the formula `may_enter(Alice, Bob)` means that Alice is allowed to enter Bob’s office, then this formula may be true during Bob’s office hours and untrue at other times.

Hence, in order to reason accurately about time in access control, the logic should reason about truth of formulas at specific times. This leads us to the basic judgment of our logic: “formula A is true at all time points in the set I ,” written $A[[I]]$.

Following prior work on security logics [19], we would like to go a step further by adding linearity to the logic for modeling state and single-use authorizations. Accordingly, we add a second judgment: “formula A is true exactly once in the set I ,” written $A[I]$. This does not mean that A holds at exactly one time point in I , but rather that the authorization implied by A must be used at one time point in the interval. For example, `may_enter(Alice, Bob)[[I]]` means that Alice may enter Bob’s office any number of times during interval I , while `may_enter(Alice, Bob)[I]` means that Alice must enter Bob’s office exactly once during interval I .

Next, in order to allow reasoning from assumptions, a feature central to all logics, we introduce a *hypothetical judgment (sequent)*. It takes the following form:

$$\Sigma, \Psi; \Gamma; \Delta \Longrightarrow A[I]$$

$\Sigma, \Psi, \Gamma, \Delta$ have the syntax listed below:

$$\begin{aligned} \Sigma &::= \cdot \mid \Sigma, x:s \\ \Psi &::= \cdot \mid \Psi, C \\ \Gamma &::= \cdot \mid \Gamma, A[[I]] \\ \Delta &::= \cdot \mid \Delta, A[I] \end{aligned}$$

Σ assigns sorts to all first-order parameters occurring in the remaining sequent. Ψ records the constraints, including upset constraints on intervals, that are assumed to hold. Γ contains assumptions that are true on specific intervals, and Δ represents assumptions that are true exactly once on specific intervals. Γ and Δ are often called *unrestricted hypotheses* and *linear hypotheses*, respectively.

The meaning of the entire sequent is: “For each solution to the constraints Ψ in the variables Σ , we can prove that A is true exactly once during interval I , using each hypothesis in Δ exactly once and each hypothesis in Γ zero or more times.”

The judgment $A[I]$ on the right side of \Longrightarrow is often called the *consequent* of the sequent. Sequents cannot have consequents of the form $A[[I]]$. This restriction is inherited from linear logic [13], but does not limit the expressiveness of the deductive system.

Affirmations. In order to model security policies issued by distinct principals, we need to reason about statements made by principals. We call such statements affirmations. Due to the hybrid nature of the logic, we have to associate time with affirmations. Accordingly, we introduce a new judgment: “during interval I it is true that principal K affirms that formula A is true,” written (K affirms A) at I .

There are two important points here. First, the phrase “ K affirms that formula A is true” is broadly construed: K may not directly state that A is true; instead, A may follow from other statements that K has made. Second, I is the interval over which the *affirmation* itself is made, not K ’s intention of the interval on which A is true. If required, the latter may be encoded within A using the @ connective. For example, suppose that Bob creates the policy “Alice may enter Bob’s office between 9 AM and 5 PM” and that this policy is valid from 2007 to 2008. Then, this fact is represented by the judgment (Bob affirms (may_enter(Alice, Bob) @ [9AM, 5PM])) at [2007, 2008]. Observe that here the validity interval I is [2007, 2008], whereas Bob intends that [9AM, 5PM] be the interval on which may_enter(Alice, Bob) is true.

Next, we add a new form of sequent to reason hypothetically about affirmations.

$$\Sigma, \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I$$

The meaning of this sequent is: “For each solution to the constraints Ψ in the variables Σ , we can prove that K affirms A exactly once during interval I , using each hypothesis in Δ exactly once and each hypothesis in Γ zero or more times.”

2.3. Constraints

Superset constraints of the form $I \supseteq I'$ are an integral part of η -logic. However, η -logic is flexible enough to permit other, application-specific constraint forms to be added as needed. We use the meta-variable C to stand for arbitrary superset or application-specific constraints.

Formally, constraints are incorporated in the proof system using the following judgment:

$$\Sigma; \Psi \models C$$

This judgment means that the constraints in Ψ entail that constraint C holds, parametrically in the constants mentioned in Σ . We do not fix the exact rules governing this judgment because we stipulate neither a concrete structure for intervals nor the full language of constraints. We expect that, in practice, this judgment would be implemented using a constraint solving procedure. The details of such a procedure would, of course, depend on the representation chosen for intervals and on the types of application-specific

constraints. However, to obtain meta-theoretic results about the logic (Section 2.5), we require the following properties.

1. (Hypothesis) $\Sigma; \Psi, C \models C$.
2. (Weakening) If $\Sigma; \Psi \models C$, then $\Sigma, \Sigma'; \Psi, \Psi' \models C$.
3. (Cut) If $\Sigma; \Psi \models C$ and $\Sigma; \Psi, C \models C'$, then $\Sigma; \Psi \models C'$.
4. (Substitution) If $\Sigma \vdash t:s$ and $\Sigma, x:s; \Psi \models C$, then $\Sigma; [t/x]\Psi \models [t/x]C$.
5. (Reflexivity) $\Sigma; \Psi \models I \supseteq I$.
6. (Transitivity) If $\Sigma; \Psi \models I \supseteq I'$ and $\Sigma; \Psi \models I' \supseteq I''$, then $\Sigma; \Psi \models I \supseteq I''$.

In the case where intervals are represented by closed intervals on the real line, such a constraint solver can be constructed in a straightforward manner by saturation.

2.4. Formulas and proof rules

Having described the basic judgments and constraints in η -logic, we now turn to the connectives allowed in formulas and the proof rules for sequents. We allow all connectives of intuitionistic linear logic, although, for the sake of brevity, we limit our discussion in this paper to only a subset. In addition, we introduce a new connective $A @ I$ to internalize the judgment $A[I]$ as a formula, include the connective $\langle K \rangle A$ (read “ K says A ”) [19, 20] to internalize the affirmation judgment, and adopt the connectives $C \dot{\supset} A$ and $C \dot{\wedge} A$ [37] to represent constraint implication and constraint conjunction, respectively.

The syntax of formulas is shown below. P denotes atomic formulas.

$$A, B ::= P \mid A \otimes B \mid A \supset B \mid A \multimap B \mid \forall x:s.A \mid A @ I \mid \langle K \rangle A \mid C \dot{\supset} A \mid C \dot{\wedge} A$$

$A \otimes B$ means that A and B are true simultaneously. We have two forms of implication: unrestricted ($A \supset B$) and linear ($A \multimap B$). They differ in that the precondition A in $A \supset B$ can be satisfied only if A can be established without the use of linear hypotheses, while there is no such restriction on the precondition of $A \multimap B$. Conversely, to prove $A \supset B$ we may use A arbitrarily many times to prove B , while A must be used exactly once in a proof of B to establish $A \multimap B$. $C \dot{\supset} A$ means that A is true if constraint C is satisfied. $C \dot{\wedge} A$ means both that constraint C is satisfied and that A is true.

Proof rules for the sequent calculus are summarized in Figure 1. γ denotes an arbitrary consequent, either $A[I]$ or (K affirms A) at I . The meanings of connectives in η -logic are described entirely by these proof rules, without any additional semantics. This ensures that the intended reading of formulas coincides with the available formal proofs, which is desirable for PCA.

The init and copy rules capture the nature of linear and unrestricted hypotheses. If we assume that formula A is true

Basic Rules

$$\frac{\Sigma; \Psi \models I \supseteq I' \quad (P \text{ atomic})}{\Sigma; \Psi; \Gamma; P[I] \Longrightarrow P[I']} \text{ init}$$

$$\frac{\Sigma; \Psi; \Gamma, A[[I]]; \Delta, A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A[[I]]; \Delta \Longrightarrow \gamma} \text{ copy}$$

$A @ I$

$$\frac{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A @ I[I']} @R$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta, A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, A @ I[I'] \Longrightarrow \gamma} @L$$

Constraints

$$\frac{\Sigma; \Psi, C; \Gamma; \Delta \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow C \dot{\supset} A[I]} \dot{\supset}R$$

$$\frac{\Sigma; \Psi \models C \quad \Sigma; \Psi; \Gamma; \Delta, A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, C \dot{\supset} A[I] \Longrightarrow \gamma} \dot{\supset}L$$

$$\frac{\Sigma; \Psi \models C \quad \Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow C \dot{\wedge} A[I]} \dot{\wedge}R$$

$$\frac{\Sigma; \Psi, C; \Gamma; \Delta, A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, C \dot{\wedge} A[I] \Longrightarrow \gamma} \dot{\wedge}L$$

Affirmation and $\langle K \rangle A$

$$\frac{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I} \text{ affirms}$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow \langle K \rangle A[I]} \langle \rangle R$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta, A[I] \Longrightarrow (K \text{ affirms } B) \text{ at } I' \quad \Sigma; \Psi \models I \supseteq I'}{\Sigma; \Psi; \Gamma; \Delta, \langle K \rangle A[I] \Longrightarrow (K \text{ affirms } B) \text{ at } I'} \langle \rangle L$$

Other Connectives

$$\frac{\Sigma; \Psi; \Gamma; \Delta_1 \Longrightarrow A[I] \quad \Sigma; \Psi; \Gamma; \Delta_2 \Longrightarrow B[I]}{\Sigma; \Psi; \Gamma; \Delta_1, \Delta_2 \Longrightarrow A \otimes B[I]} \otimes R$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta, A[I], B[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, A \otimes B[I] \Longrightarrow \gamma} \otimes L$$

$$\frac{\Sigma, i:\text{interval}; \Psi, I \supseteq i; \Gamma; \Delta, A[i] \Longrightarrow B[i]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A \multimap B[I]} \multimap R$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta_1 \Longrightarrow A[I'] \quad \Sigma; \Psi \models I \supseteq I' \quad \Sigma; \Psi; \Gamma; \Delta_2, B[I'] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta_1, \Delta_2, A \multimap B[I] \Longrightarrow \gamma} \multimap L$$

$$\frac{\Sigma, i:\text{interval}; \Psi, I \supseteq i; \Gamma, A[[i]]; \Delta \Longrightarrow B[i]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A \supset B[I]} \supset R$$

$$\frac{\Sigma; \Psi; \Gamma; \cdot \Longrightarrow A[I'] \quad \Sigma; \Psi \models I \supseteq I' \quad \Sigma; \Psi; \Gamma; \Delta, B[I'] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, A \supset B[I] \Longrightarrow \gamma} \supset L$$

$$\frac{\Sigma, a:s; \Psi; \Gamma; \Delta \Longrightarrow [a/x]A[I]}{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow \forall x:s.A[I]} \forall R$$

$$\frac{\Sigma; \Psi; \Gamma; \Delta, [t/x]A[I] \Longrightarrow \gamma \quad \Sigma \vdash t:s}{\Sigma; \Psi; \Gamma; \Delta, \forall x:s.A[I] \Longrightarrow \gamma} \forall L$$

Figure 1. Sequent calculus for η -logic

once during interval I , and if $I \supseteq I'$, then we should certainly be able to conclude that A may be true once during the interval I' . For atomic formulas, this is captured by the init rule; for others, we prove it as a theorem (Theorem 2). The init rule also highlights the interaction between linearity, time, and constraints: its premise contains a constraint,

and the fact that no other linear hypothesis besides $P[I]$ is allowed to occur captures linearity. The copy rule permits copying of an unrestricted hypothesis $A[[I]]$ into the set of linear hypotheses. This may be repeated, thus allowing the unrestricted hypothesis to be used multiple times.

The remaining rules (with the exception of affirms) are

related to the logic's connectives. Each rule is classified as either right or left, depending on whether it acts on the right side or the left side of \implies . We start with the new connective: $A @ I$.

$A @ I$ captures the essence of the judgment $A[I]$ as a formula. This permits us to associate time intervals with formulas nested inside other formulas. The right rule, $@R$, means that $A[I]$ entails $A @ I[I']$. The left rule, $@L$, states that the assumption $A @ I[I']$ is stronger than the assumption $A[I]$. Together they imply that, as judgments, $A[I]$ and $A @ I[I']$ entail each other. This is intuitive: if a formula A is true during interval I , then this fact is true over all intervals I' . Or equivalently, once the truth of a formula has been qualified by an interval, a subsequent qualification is meaningless.

Next, we consider constraint implication. According to the right rule, $\dot{\supset}R$, to establish the judgment $C \dot{\supset} A[I]$, it is sufficient to establish that A is true during I while assuming that constraint C holds. Dually, if we assume the judgment $C \dot{\supset} A[I]$ and can show that the constraint C is satisfied, then we are justified in assuming that A is true during I . This is captured by the left rule, $\dot{\supset}L$.

η -logic also includes conjunctions of constraints and formulas. The right rule, $\wedge R$, states that the judgment $C \wedge A[I]$ can be established whenever it can be shown that both constraint C holds and A is true during I . The left rule, $\wedge L$, states that we can simultaneously project out the components of a constraint conjunction: if we assume that $C \wedge A[I]$, then we may also assume that constraint C holds and that A is true during I .

Next, we examine affirmation. The affirms rule relates affirmation to truth. It states that if it is provable that formula A is true during interval I , then it is provable that every principal affirms its truth during interval I . This is based on the idea that a proof is irrefutable evidence; if A has a proof, then every principal must be willing to affirm A .

The connective $\langle K \rangle A$ (read “ K says A ”) internalizes affirmation as a formula. Its right rule, $\langle \rangle R$, means that the judgment $(\langle K \rangle A)[I]$ holds whenever $(K$ affirms $A)$ at I holds. The left rule, $\langle \rangle L$, means that if we are trying to establish that K affirms B during I' , and we know both K says A during I and $I \supseteq I'$, then we are justified in assuming that A is true during I . This rule captures the idea that principals are accountable for their statements; having stated A , K cannot refute it, and hence, while reasoning about another affirmation by K , we can assume A . It is instructive to observe the interaction between time and affirmation in this rule.

Finally, we describe the rules for connectives borrowed from linear logic: \otimes , \multimap , \supset , and \forall . Although these rules may appear similar to the corresponding rules in linear logic (without time), the meanings of the connectives

must be reinterpreted because truth is always qualified with time in η -logic. The presence of time opens the possibility of choosing from many different kinds of rules, with each choice resulting in a different interaction between the connectives and $@$. For instance, our rules imply that $@$ distributes over \otimes — that $(A \otimes B) @ I$ is equivalent to $(A @ I) \otimes (B @ I)$. However, this choice is not forced, and one may conceive logics that do not validate this equivalence. The proof rules shown here describe what we believe to be an elegant, useful, and simple possibility.

$\otimes R$, the right rule for \otimes , states that in order to show that $A \otimes B$ is true on I , it suffices to partition the linear hypotheses disjointly into two parts, using one part to establish that A holds on I and the other to show that B holds on I . The left rule, $\otimes L$, is dual, stating that the assumption $A \otimes B$ on interval I is stronger than the pair of assumptions A, B , both on the interval I . Together with the rules for $@$, these rules imply the equivalence mentioned earlier.

$\multimap R$, the right rule for \multimap means that in order to establish that $A \multimap B$ holds on interval I , it suffices to show that for every interval i such that $I \supseteq i$, $B[i]$ follows from the linear hypothesis $A[i]$. The left rule, $\multimap L$, is dual, stating that if $A \multimap B$ is assumed to hold on I and A holds on any smaller interval I' , then B holds on I' . Together these mean that $(A \multimap B) @ I$ represents a method of obtaining B from A on any subset of I .

The $\supset R$ rule is similar to $\multimap R$, except that in this case A is assumed to be unrestricted. Correspondingly, in the left rule $\supset L$, one must establish A without any linear hypotheses.

We can establish the formula $\forall x:s.A$ if we can establish $[a/x]A$ for every fresh constant a of sort s . This is captured by the right rule $\forall R$. The left rule $\forall L$ states that if we assume $\forall x:s.A$, then we can also assume $[t/x]A$ for any term t of sort s .

This completes our presentation of the proof rules of the sequent calculus. We now turn to the meta-theory of η -logic.

2.5. Meta-theory

Meta-theoretic properties are important for a logic of authorization because they not only provide assurance of a strong foundation for the logic, but are also useful in analysis of policies. Cut elimination, for example, implies that all proofs can be normalized, i.e., reduced to a canonical form. This canonical form often provides far more insight into the justifications for access than the original proof does.

In our logic, meta-theoretic properties are important from yet another perspective. Since connectives are described entirely by the rules of the sequent calculus, it is absolutely essential that the basic meaning of hypothetical judgments (sequents) be respected by the rules. Formally,

this is expressed by two properties: admissibility of cut and identity. Admissibility of cut states that if we can establish a judgment such as $A[I]$, and assuming this judgment we can establish a second judgment, then the second judgment can be established directly. The identity principal states that whenever we assume a judgment, we can conclude it. We prove both properties for our logic. To establish admissibility of cut, it must be stated in a more general form.

Theorem 1 (Admissibility of Cut).

1. If $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]$ and $\Sigma; \Psi; \Gamma; \Delta', A[I] \Longrightarrow \gamma$, then $\Sigma; \Psi; \Gamma; \Delta', \Delta \Longrightarrow \gamma$.
2. If $\Sigma; \Psi; \Gamma; \cdot \Longrightarrow A[I]$ and $\Sigma; \Psi; \Gamma, A[I]; \Delta' \Longrightarrow \gamma$, then $\Sigma; \Psi; \Gamma; \Delta' \Longrightarrow \gamma$.
3. If $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I$ and $\Sigma; \Psi; \Gamma; \Delta', A[I] \Longrightarrow (K \text{ affirms } C) \text{ at } I'$ and $\Sigma; \Psi \models I \supseteq I'$, then $\Sigma; \Psi; \Gamma; \Delta', \Delta \Longrightarrow (K \text{ affirms } C) \text{ at } I'$.

Theorem 2 (Identity). For any proposition A , $\Sigma; \Psi; \Gamma; A[I] \Longrightarrow A[I']$ if $\Sigma; \Psi \models I \supseteq I'$.

Cut elimination usually refers to the explicit elimination of cut as a rule of inference from the sequent calculus. It follows by a simple structural induction from the admissibility of cut, and is therefore omitted here.

In a hybrid logic like η -logic, we expect another important property: if we can establish $A[I]$, then we should be able to establish $A[I']$ for every subset I' of I . This property, called subsumption, is formally captured by the following theorem.

Theorem 3 (Subsumption). If $\Sigma; \Psi \models I \supseteq I'$, then the following hold:

1. If $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]$, then $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I']$.
2. If $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I$, then $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I'$.

We now state some simple theorems that hold in the logic. Equally important are properties that cannot be established in their full generality. We write $\vdash A$ if, for any Σ, Ψ , and I'' , it is the case that $\Sigma; \Psi; \cdot \Longrightarrow A[I'']$, and write $\not\vdash A$ otherwise. $A \equiv B$ is an abbreviation for $(A \multimap B) \otimes (B \multimap A)$.

1. $\not\vdash (A @ I) \multimap (A @ I')$
2. $\vdash (I \supseteq I') \dot{\supset} ((A @ I) \multimap (A @ I'))$
3. $\vdash (A @ I @ I') \equiv (A @ I)$
4. $\vdash ((A \otimes B) @ I) \equiv ((A @ I) \otimes (B @ I))$
5. $\not\vdash A[]$

The first property states that, in general, $A @ I$ does not imply $A @ I'$. In the special case where I is a superset of I' , this is true (second property). The third property captures

the nature of nested $@$ connectives: $A @ I @ I'$ and $A @ I$ are equivalent. The fourth property implies that $@$ distributes over \otimes . The last property states consistency — not every formula is provable a priori in the logic.

The says connective $\langle K \rangle A$ is similar to a lax modality [17]. It satisfies the following properties.

1. $\vdash A \multimap \langle K \rangle A$
2. $\vdash (\langle K \rangle \langle K \rangle A) \multimap \langle K \rangle A$
3. $\vdash (\langle K \rangle (A \multimap B)) \multimap ((\langle K \rangle A) \multimap \langle K \rangle B)$
4. $\not\vdash (\langle K \rangle A) \multimap A$

As a general design decision, we have kept the interaction between temporal constraints and logical reasoning as simple as possible. In particular, we do not permit splitting of intervals into sub-intervals during logical reasoning. For example, even if $I \cup I' = I''$ we can not prove in general that $A @ I$ and $A @ I'$ imply $A @ I''$. For proof-carrying authorization (discussed in the next section), this means in order to demonstrate continuous right to access a resource over a given interval there must be a uniform proof over the whole interval, unless special policy axioms are introduced. The logic can easily be generalized to permit splitting of intervals, but theorem proving becomes significantly more difficult. Jia [24] provides an analysis of this trade-off in the setting of reasoning about imperative programs using a heap.

3. Proof-carrying authorization with η -logic

In this section, we describe applications of η -logic to PCA. The main merit of using η -logic for PCA is that temporal validity of policies and credentials gets reflected in formulas of the logic, thus bringing the formalized policies closer to their intended meaning. We review the Grey system [5, 6] in Section 3.1 and use it as an example to illustrate our PCA approach in Section 3.2. In Section 3.3, we comment on the feasibility of using η -logic in PCA. Finally, we formalize some of our claims about enforcement in Section 3.4.

3.1. Review of the Grey system

The Grey system is an architecture for universal access control using proof-carrying authorization with smartphones. The Grey testbed is an implementation of keyless access control on office doors and computers, developed and currently deployed on one floor in the Collaborative Innovation Center at Carnegie Mellon University. Each office door is equipped with a processor that runs a proof-checking engine based on a logical framework. The processor controls an electronic relay which can unlock the door.

Enforcement of access control follows the standard PCA approach: a person desiring access to an office uses her cell-phone to communicate with the office’s door, sending it a proof that she is allowed access. This proof is checked by the proof-checking engine in the door, and if the proof is correct, the processor unlocks the door through the relay.

Two simple policies in Grey are the following:

1. A person may enter her own office.
2. A person may enter an office not belonging to her, if authorized to do so by the owner of the office.

In addition to policies, authorization in Grey relies on credentials issued by individual users that authorize other users to enter their offices. They are used in conjunction with the second policy. Physically, these credentials are digitally signed X.509 certificates. For pragmatic reasons, most of these credentials are not valid forever because an individual usually does not want to allow another person access to her office indefinitely.

Each policy statement and each available credential is converted to a formula in Grey’s logic. An individual wanting access must not only provide the door with a proof, but also any credentials used in the proof that the door may be unaware of. In addition to checking the proof, the door also checks the new credentials. If both checks succeed, the door opens. Otherwise, it does not.

Grey’s current logic is oblivious to time. As a result, the temporal validity of credential is ignored when the credential is imported into the logic. For example, suppose Bob signs the credential “Allow Alice to enter my door (valid from 1/1/08 to 1/31/08).” If the predicate $\text{may_enter}(K_1, K_2)$ means that K_1 is allowed to enter K_2 ’s office, then this credential may be imported into Grey’s logic as the formula $\langle \text{Bob} \rangle \text{may_enter}(\text{Alice}, \text{Bob})$. The validity bound of the credential is ignored in the logic. Policies are treated similarly — their validity, if any, is ignored. Consequently, proofs are ignorant of time, and it is possible to obtain a seemingly correct proof in the logic depending on formulas derived from expired credentials.

In order to rectify this problem and correctly enforce the time bounds in credentials, Grey uses an extra-logical mechanism. In addition to checking that a submitted proof and credentials are correct, a door also checks that all credentials used in the proof are valid *at the time of access*. Although secure and efficient in practice, this method divorces time from the logic, making reasoning in the logic inaccurate with respect to time. In particular, proof construction has to be augmented with a similar external time check. Otherwise, correct but expired proofs may be constructed. Furthermore, any meta-level analysis of the policies using the logic will be inaccurate with respect to time.

3.2. Grey in η -logic

In η -logic, we can model time-bounded credentials accurately. We illustrate this using policies from the Grey system. As before, let the predicate $\text{may_enter}(K_1, K_2)$ mean that K_1 is allowed to enter K_2 ’s office. We assume the existence of an administrating principal, admin , who dictates all policies. For this example and all subsequent ones, we assume that time is represented by points on the real line, and intervals in the logic are intervals on the real line.

To open K_2 ’s door at time t , K_1 must submit a proof showing that the judgment $\langle \text{admin} \rangle \text{may_enter}(K_1, K_2)[t, t]$ is derivable from the available policies and credentials. $[t, t]$ represents the point interval containing only the time point t . Observe that the judgment that must be established to gain access directly incorporates time. This is in sharp contrast to Grey’s existing approach, where time is external to the logic.

Grey’s policies described earlier can be imported as the following unrestricted hypotheses in η -logic.

1. $\langle \text{admin} \rangle \forall K. \text{may_enter}(K, K)[(-\infty, \infty)]$
2. $\langle \text{admin} \rangle \forall K_1. \forall K_2. \langle K_2 \rangle \text{may_enter}(K_1, K_2) \rightarrow \text{may_enter}(K_1, K_2)[(-\infty, \infty)]$

Here we have assumed that both policies are valid indefinitely, i.e., on the interval $(-\infty, \infty)$. If the policies are valid for only a finite duration of time, one may replace $(-\infty, \infty)$ with the appropriate interval.

A critical observation is that we assume that these formulas are unrestricted hypotheses because they may be used many times. This does not apply to credentials issued by individuals to allow others to enter their offices. For example, Bob may allow Alice to enter his office *once* between 1/1/08 and 1/31/08 by issuing a certificate that is imported as the linear hypothesis:

3. $\langle \text{Bob} \rangle \text{may_enter}(\text{Alice}, \text{Bob})[1/1/08, 1/31/08]$

It is instructive to check that using the unrestricted hypotheses (1) and (2) and the linear hypothesis (3), it is possible to derive $\langle \text{admin} \rangle \text{may_enter}(\text{Alice}, \text{Bob})[t, t]$ for any t in the time interval $[1/1/08, 1/31/08]$. Also, it is impossible to derive the same judgment if t does not lie in this interval. Thus, qualifying formulas explicitly with intervals on which they are true makes proof construction in the logic accurate with respect to time bounds on credentials.

3.3. Implementing PCA with η -logic

As described above, allowing explicit time in a logic bridges the gap between time-dependent credentials and their representation in the logic. The question then is

whether this approach offers any advantages over traditional implementations of PCA.

The primary issue is efficiency. At first, one might think that adding time to the logic would slow proof-checking. While a comprehensive assessment of the efficiency of proof-checking can only be made with a real implementation, we show in Section 3.4 that a reasonable fragment of the logic (namely, one in which there are no nested $@$ or constraint connectives), can be implemented using the same method that Grey uses to enforce time-dependence of credentials: proof-checking and proof construction are done in oblivion to time, and validity of certificates at the time of access is ascertained separately. This fragment is large enough to express all policies of Grey, and other existing PCA based systems.

Thus existing PCA systems can be implemented in η -logic without loss of efficiency. At the same time there are several merits in making time explicit in the logic. First, policies and credentials are reflected more accurately in the logic. They therefore become amenable to more accurate meta-level policy analysis, such as an analysis for security loopholes. Second, by leveraging the existing constraint-solving mechanism, one can model complex policies, policies that are intractable in previously proposed logics. Examples in Section 4 include such policies. Third, with time-aware formulas, one cannot, even accidentally, construct a proof that is invalid due to a time-dependence. This reduces the risk of unanticipated access denials.

We anticipate new challenges if PCA is implemented using a fragment of η -logic larger than the one described above. An important issue in proof search is certificate chain discovery, i.e., the problem of finding credentials that are relevant for a proof. In a time-aware logic, this problem is exacerbated, since this process has to incorporate temporal validity of certificates. However, there is a trade-off here: at the cost of more work, the final proof is guaranteed to be accurate. Alternatively, one may choose to ignore time during proof search. In that case, certificate chain discovery would revert to its usual complexity (and time-dependent inaccuracy).

An essential component that must be built into any realistic implementation of η -logic is a constraint solver. For simple constraints such as $I \supseteq I'$ that we have seen so far, this appears to be relatively straightforward. Furthermore, most policies arising in practice do not require parameters in constraints. This trivializes the constraint solving problem to checking containment over ground intervals. Even if one wished to be more ambitious by allowing other kinds of constraints for use in policies, previous work in constraint logic programming suggests that a large number of useful constraint domains are tractable in practice (see [23] for a survey).

An interesting, open problem in implementing PCA with

η -logic is the treatment of linearity. Since linear hypotheses and the corresponding credentials must be consumed only once, a mechanism for tracking their use is required. If all linear credentials are maintained in a central database, this is relatively straightforward. It is less clear, however, whether there is a uniform way of doing this in a completely distributed setting. Some initial ideas using contract signing protocols have been described earlier [10].

3.4. Enforcement for a fragment of η -logic

The objective of this section is to show that Grey's method of checking credential validity at the time of request as a separate step after proof-checking can also be used for the fragment of η -logic without the $@$, $\dot{\supset}$, and $\dot{\wedge}$ connectives. This fragment does not preclude intervals in top-level judgments such as $A[I]$ and $A[[I]]$. It covers all systems in which time is used only to bound the validity of credentials, but not inside the text of credentials, including all policies of the Grey system.

In order to formally describe our result we need a logic without time which is otherwise similar to η -logic. We choose the logic of [19], since our logic is derived from it. For the lack of a better name, we call this logic ζ -logic (ζ being the predecessor of η in the Greek alphabet). Limitations of space do not permit us to describe ζ -logic in detail, but it may be understood as the simplification of η -logic obtained by erasing intervals and constraints from formulas, judgments, sequents, and proof rules. The uninitiated reader may skip this section without affecting readability of the remaining paper.

Let F denote formulas which do not contain the $@$, $\dot{\supset}$, and $\dot{\wedge}$ connectives. Such formulas are in the syntax of ζ -logic. Let Θ and Λ denote multisets of such formulas, representing unrestricted hypotheses and linear hypotheses in ζ -logic, respectively. Let \vec{I} denote a list of ground intervals. Furthermore, if $\Theta = F_1, \dots, F_n$, and $\vec{I} = I_1, \dots, I_n$, let $\Theta[[\vec{I}]]$ denote the set of unrestricted hypotheses $F_1[[I_1]], \dots, F_n[[I_n]]$ in η -logic. Define $\Lambda[[\vec{I}]]$ similarly.

All sequents in this fragment of η -logic have one of the forms $\Sigma; \Psi; \Theta[[\vec{I}]]; \Lambda[[\vec{I}]] \Longrightarrow F[I'']$ or $\Sigma; \Psi; \Theta[[\vec{I}]]; \Lambda[[\vec{I}]] \Longrightarrow (K \text{ affirms } F) \text{ at } I''$.

Our idea for implementing PCA with this fragment of η -logic is the following. Whenever a principal needs to prove $\Sigma; \Psi; \Theta[[\vec{I}]]; \Lambda[[\vec{I}]] \Longrightarrow F[I'']$, she instead proves that $\Sigma; \Theta; \Lambda \Longrightarrow F$ in ζ -logic. The proof checker verifies this proof in ζ -logic, and checks that each interval in \vec{I} and \vec{I}' is a superset of I'' . As the following theorem shows, the success of these two checks implies that the original sequent is provable in η -logic.

(A priori, this result was not obvious to us because intervals mentioned in the last sequent of a proof interact with

subformulas in other sequents of the proof. It seemed entirely possible that some subtle consequence of these interactions would not be captured by simply checking that each interval in \vec{I} and \vec{I}' is a superset of I'' .)

Theorem 4. *Suppose $\Sigma; \Psi \models I''' \supseteq I''$ for each $I''' \in \vec{I}$ and for each $I''' \in \vec{I}'$. Then,*

1. *If $\Sigma; \Theta; \Lambda \implies F$ in ζ -logic, then $\Sigma; \Psi; \Theta[\vec{I}]; \Lambda[\vec{I}'] \implies F[I'']$ in η -logic.*
2. *If $\Sigma; \Theta; \Lambda \implies K$ affirms F in ζ -logic, then $\Sigma; \Psi; \Theta[\vec{I}]; \Lambda[\vec{I}'] \implies (K \text{ affirms } F)$ at I'' in η -logic.*

Thus, on the fragment without $\textcircled{\wedge}$, $\dot{\supset}$, and $\hat{\wedge}$, proof-checking in a logic without time, together with simple containment checking for intervals *soundly* approximates proof-checking in η -logic. One might also expect the converse to hold, namely that whenever $\Sigma; \Psi; \Theta[\vec{I}]; \Lambda[\vec{I}'] \implies F[I'']$ holds in η -logic, $\Sigma; \Theta; \Lambda \implies F$ holds in ζ -logic and for each interval $I''' \in \vec{I}$ and each $I''' \in \vec{I}'$, $\Sigma; \Psi \models I''' \supseteq I''$. This is partially correct: given that the η -logic sequent is provable, the former holds as the following theorem shows, but the latter may not. The reason is quite straightforward: the consequent of the sequent may not depend on some assumptions in Θ , and the intervals associated with such assumptions may have no relation to I'' .

Theorem 5.

1. *If $\Sigma; \Psi; \Theta[\vec{I}]; \Lambda[\vec{I}'] \implies F[I'']$, then $\Sigma; \Theta; \Lambda \implies F$ in ζ -logic.*
2. *If $\Sigma; \Psi; \Theta[\vec{I}]; \Lambda[\vec{I}'] \implies (K \text{ affirms } F)$ at I'' , then $\Sigma; \Theta; \Lambda \implies K \text{ affirms } F$ in ζ -logic.*

4. Expressiveness of η -logic: More examples

Besides modeling time-bounded credentials, η -logic, through its combination of explicit time and constraints, can also be used to express very complicated policies. We illustrate this expressiveness through two hypothetical examples. The first example describes the policies of a homework assignment administration system at a university. In addition to time, this example uses linearity to model changes of state. The second example describes the policies of a peer review publication process.

A homework assignment administration system. We consider the policies of a hypothetical homework administration system in a university. These policies allow professors to create assignments for the courses they teach and adjust their release and due dates. Time is used explicitly to encode the release and due dates of each assignment. Students can view an assignment *after* the release date and submit it *before* the due date. Modeling this policy creates

complex interactions between time and authorization that cannot be captured without either a connective like $\textcircled{\wedge}$ or constraints.

We use the meta-variable A to denote assignments, C for courses, P for professors, and S for students. The predicates (with their intuitive meanings) and policies used in this example are summarized in Figure 2. As a syntactic convention, we assume that \otimes , \multimap , and \supset are right associative and that the binding precedences are, in decreasing order: $\langle \rangle$; $\textcircled{\wedge}$; \otimes ; \multimap and \supset ; \forall . We write $t \in I$ as an abbreviation for $I \supseteq [t, t]$, and $t \geq t'$ as an abbreviation for $t \in [t', \infty)$.

As may be expected, all policy rules are unrestricted hypotheses that are valid forever. This is indicated by the annotation $\llbracket (-\infty, \infty) \rrbracket$ on each policy rule.

We assume an administrating principal, *admin*. At the beginning of each semester, this principal issues credentials to students registered for courses and professors teaching courses. These must be presented later (perhaps many times) to view, submit, and change assignments. As a result, they are assumed to be unrestricted hypotheses. They have the logical forms $\langle \text{admin} \rangle \text{is_student}(S, C) \llbracket \text{Sem} \rrbracket$ and $\langle \text{admin} \rangle \text{is_professor}(P, C) \llbracket \text{Sem} \rrbracket$ respectively, where *Sem* denotes the semester under consideration.

A professor P can create an assignment A in a course C by issuing a credential stating $\langle P \rangle \text{is_assignment}(A, C) [t_r, t_d]$. The time points t_r and t_d stand for the release and due dates of the assignment, respectively. $[t_r, t_d]$ denotes the closed interval between these time points. We require that such credentials be linear hypotheses. If instead they were unrestricted, then there would be no logical mechanism to change the release and due dates after creating an assignment.

To view an assignment A in course C at time t , a student S must be able to prove the judgment $\langle \text{admin} \rangle \text{may_view}(S, A, C) [t, t]$. The policy rule named *view* allows students to do this. We assume an implicit universal quantification over the variables S, A, C, t, P, t_r , and t_d . Intuitively, this rule states that a student S may view an assignment A in course C at time t by issuing a credential $\langle S \rangle \text{request_view}(A, C)$ valid at the time of request, $[t, t]$, if the following can be established:

1. $\langle \text{admin} \rangle \text{is_student}(S, C) \textcircled{\wedge} [t, t]$, i.e., the student is registered for the course at the time of request. To establish this, the student must use the credential she received from *admin* at the beginning of the semester.
2. $\langle P \rangle \text{is_assignment}(A, C) \textcircled{\wedge} [t_r, t_d]$, i.e., a professor P states that A is an assignment of course C with release date t_r and due date t_d .
3. $\langle \text{admin} \rangle \text{is_professor}(P, C) \textcircled{\wedge} [t_r, t_d]$, i.e., P is a professor teaching the course C for the entire duration of the assignment. This can be established using the credential issued by *admin* to the professor.

Predicates

$\text{request_view}(A, C)$	A request to view assignment A of course C .
$\text{request_submit}(A, C)$	A request to submit answers for assignment A of course C .
$\text{is_professor}(P, C)$	P is a professor for course C .
$\text{is_student}(S, C)$	S is a student enrolled in course C .
$\text{is_assignment}(A, C)$	A is an assignment for the students in course C .
$\text{may_view}(S, A, C)$	S may view assignment A of course C .
$\text{may_submit}(S, A, C)$	S may submit answers for assignment A of course C .
$\text{change_date}(A, C, t'_r, t'_d)$	A request to change the release and due dates for assignment A of course C to t'_r and t'_d , respectively.

Policies

$\text{view} : (\langle S \rangle \text{request_view}(A, C) @ [t, t] \multimap$ $\langle \text{admin} \rangle \text{is_student}(S, C) @ [t, t] \supset$ $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d] \multimap$ $\langle \text{admin} \rangle \text{is_professor}(P, C) @ [t_r, t_d] \supset$ $(t \geq t_r) \dot{\supset}$ $\langle \text{admin} \rangle \text{may_view}(S, A, C) @ [t, t] \otimes$ $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d] [(-\infty, \infty)])]$	$\text{submit} : (\langle S \rangle \text{request_submit}(A, C) @ [t, t] \multimap$ $\langle \text{admin} \rangle \text{is_student}(S, C) @ [t, t] \supset$ $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d] \multimap$ $\langle \text{admin} \rangle \text{is_professor}(P, C) @ [t_r, t_d] \supset$ $(t \in [t_r, t_d]) \dot{\supset}$ $\langle \text{admin} \rangle \text{may_submit}(S, A, C) @ [t, t] \otimes$ $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d] [(-\infty, \infty)])]$
$\text{change} : (\langle P \rangle \text{change_date}(A, C, t'_r, t'_d) \multimap$ $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d] \multimap$ $\langle \text{admin} \rangle \text{is_professor}(P, C) \dot{\supset}$ $\langle P \rangle \text{is_assignment}(A, C) @ [t'_r, t'_d] [(-\infty, \infty)])]$	

Figure 2. Predicates and policies for a homework assignment administration system

- $t \geq t_r$, i.e., the time of request is after the release of the assignment. This preempts attempts to read the assignment before it is officially released.

If each of these four conditions are satisfied, then the student may view the assignment. There are two important observations to be made here. First, the linear hypothesis $\langle P \rangle \text{is_assignment}(A, C) @ [t_r, t_d]$ consumed in condition 2 is regenerated at the end. Second, explicit time is crucial for modeling the constraint $t \geq t_r$. Such a policy rule cannot be modeled using only time bounds on credentials.

Similarly, the submit policy rule allows a student S to submit an assignment between its release and due dates by issuing a credential of the form $\langle S \rangle \text{request_submit}(A, C)[t, t]$. In this case the objective is to establish that $\langle \text{admin} \rangle \text{may_submit}(S, A, C) @ [t, t]$, where t is the time at which the submission is made.

Our final policy rule, *change*, illustrates the use of linearity in modeling change of state. It allows a professor P to change the release and due dates of an assignment A in a course C he is teaching by issuing the credential $\langle P \rangle \text{change_date}(A, C, t'_r, t'_d)$, where t'_r and t'_d are the new release and due dates of the assignment. The policy *consumes* the earlier hypothesis defining the release and due dates of the assignment and replaces it with a new

one. For this to work properly, it is essential that such hypotheses be linear, not unrestricted. Failure to ensure this would result in two hypotheses defining the release and due dates of the same assignment after application of the rule.

A peer review publication process. We further illustrate the expressiveness of our logic by describing the policies of a hypothetical peer review and publication process of an academic journal. This example differs slightly from the previous example in that the policies are not fixed. Instead, they are created by principals using templates.

We use the meta-variable A to range over articles considered for publication, R and K for reviewers, J for journals, and E for editors. The predicates and policies used in this example are summarized in Figure 3. We stipulate that each journal J appoint an editor E during time period I by issuing the credential $\langle J \rangle \text{is_editor}(E, J)[I]$. The editor E can then declare R a reviewer for article A time t onward by issuing the credential $\langle E \rangle \text{is_reviewer}(R, A, J)[[t, \infty)]$.

In addition, E can start accepting reviews by issuing a credential that establishes the accept policy. While issuing the credential, the editor should instantiate I_E to the interval over which reviews may be accepted. All variables other than E and I_E are assumed to be universally quantified. Once established, the policy allows an appointed reviewer

Predicates

$\text{is_approved}(A, K, J)$	Article A is approved by principal K for publication in journal J .
$\text{is_reviewer}(R, A, J)$	R is the reviewer for article A submitted to journal J .
$\text{is_editor}(E, J)$	E is an editor for journal J .
$\text{is_published}(A, J)$	Article A is published in journal J .

Policies

$\text{approve} : \langle E \rangle (\langle R \rangle \text{is_approved}(A, R, J) @ [t_a, t_a] \multimap \text{is_reviewer}(R, A, J) @ [t_a, t_a] \supset (t_a \in I_E) \dot{\supset} \text{is_approved}(A, E, J) @ [t_a, \infty)) [(-\infty, \infty)]$	$\text{publish} : \langle J \rangle (\langle E \rangle \text{is_approved}(A, E, J) @ [t_a, t'_a] \multimap \text{is_editor}(E, J) @ [t_a, t_a] \supset (t_a \in I_J) \dot{\supset} \text{is_published}(A, J) @ [t_a, \infty)) [(-\infty, \infty)]$
---	---

Figure 3. Predicates and policies for a peer review publication process

R to submit a review on article A at time t_a by signing the credential $\langle R \rangle \text{is_approved}(A, R, J)[t_a, t_a]$. If $t_a \in I_E$ the policy can be used to conclude that the editor considers the article approved.

In an analogous manner, each journal J can establish a publishing policy by issuing a credential following the form of publish. In issuing this credential, I_J should be instantiated to the interval during which articles are accepted for publication. All variables other than J and I_J are assumed to be universally quantified. Once established, the policy states that if an editor E says at time t_a that an article A has been approved, and t_a is in I_J , then the article is considered published from time t_a onward.

5. Conclusion

This paper has presented a logic that combines time, linearity, hybrid worlds, and authorization in a novel way. Our proof-theoretic approach resulted in a clean meta-theory. Among other properties, we established cut elimination. We also showed that a reasonably expressive fragment of our logic can be enforced in a PCA architecture in a straightforward manner. Through examples, we illustrated the expressiveness of the logic and demonstrated scenarios which cannot be modeled in earlier proposals.

An important topic that remains open is the analysis of policies written in the logic. We expect that work from prior logics, particularly affirmation flow analysis theorems [20], will carry over to η -logic. It will be interesting to study how these theorems interact with time.

References

- [1] M. Abadi. Access control in a core calculus of dependency. In *ICFP '06: Proceedings of the eleventh ACM SIGPLAN international conference on Functional programming*, pages 263–273, New York, NY, USA, 2006. ACM Press.
- [2] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.
- [3] A. W. Appel and E. W. Felten. Proof-carrying authentication. In G. Tsudik, editor, *Proceedings of the 6th Conference on Computer and Communications Security*, pages 52–62, Singapore, November 1999. ACM Press.
- [4] L. Bauer. *Access Control for the Web via Proof-Carrying Authorization*. PhD thesis, Princeton University, November 2003.
- [5] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In *Information Security: 8th International Conference (ISC '05)*, Lecture Notes in Computer Science, pages 431–445, September 2005.
- [6] L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 Symposium on Security and Privacy*, pages 81–95, May 2005.
- [7] M. Y. Becker, C. Fournet, and A. D. Gordon. Design and semantics of a decentralized authorization language. In *20th IEEE Computer Security Foundations Symposium*, pages 3–15, 2007.
- [8] M. Y. Becker and P. Sewell. Cassandra: Flexible trust management applied to health records. In *Proceedings of IEEE Computer Security Foundations Workshop*, pages 139–154, 2004.
- [9] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1):71–127, 2003.
- [10] K. D. Bowers, L. Bauer, D. Garg, F. Pfenning, and M. K. Reiter. Consumable credentials in logic-based access-control systems. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS '07)*, San Diego, California, February 2007.
- [11] T. Braüner and V. de Paiva. Towards constructive hybrid logic. In *Electronic Proceedings of Methods for Modalities 3 (M4M3)*, 2003.
- [12] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. Audit-based compliance control. *Int. J. Inf. Secur.*, 6(2):133–151, 2007.

- [13] B.-Y. E. Chang, K. Chaudhuri, and F. Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, 2003.
- [14] J. DeTreville. Binder, a logic-based security language. In M. Abadi and S. Bellare, editors, *Proceedings of the 2002 Symposium on Security and Privacy (S&P'02)*, pages 105–113, Berkeley, California, May 2002. IEEE Computer Society Press.
- [15] H. DeYoung, D. Garg, and F. Pfenning. An authorization logic with explicit time. Technical Report CMU-CS-07-166, Computer Science Department, Carnegie Mellon University, December 2007.
- [16] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*. The MIT Press, 1990.
- [17] M. Fairtlough and M. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, Aug. 1997.
- [18] T. Frühwirth. Temporal annotated constraint logic programming. *Journal of Symbolic Computation*, 22(5-6):555–583, 1996.
- [19] D. Garg, L. Bauer, K. Bowers, F. Pfenning, and M. Reiter. A linear logic of affirmation and knowledge. In D. Gollman, J. Meier, and A. Sabelfeld, editors, *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS '06)*, pages 297–312, Hamburg, Germany, September 2006. Springer LNCS 4189.
- [20] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In J. Guttman, editor, *Proceedings of the 19th Computer Security Foundations Workshop (CSFW '06)*, pages 283–293, Venice, Italy, July 2006. IEEE Computer Society Press.
- [21] G. 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] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [23] J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
- [24] L. Jia. *Linear Logic and Imperative Programming*. PhD thesis, Department of Computer Science, Princeton University, November 2007.
- [25] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of IEEE Symposium on Security and Privacy*, 2001.
- [26] M. I. Kanovich, M. Okada, and A. Scedrov. Specifying real-time finite-state systems in linear logic. In *2nd International Workshop on Constraint Programming for Time-Critical Applications and Multi-Agent Systems (COTIC '98)*, volume 16 of *Electronic Notes in Theoretical Computer Science*, pages 42–59, Nice, France, September 1998.
- [27] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.
- [28] C. Lesniewski-Laas, B. Ford, J. Strauss, M. F. Kaashoek, and R. Morris. Alpaca: extensible authorization for distributed services. In *14th ACM Conference on Computer and Communications Security*, 2007. To appear.
- [29] N. Li, B. N. Grosz, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.
- [30] N. Li and J. C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73. Springer-Verlag, 2003.
- [31] N. Li, J. C. Mitchell, and W. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, 2002.
- [32] P. 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.
- [33] B. Moszkowski and Z. Manna. Reasoning in interval temporal logic. In E. Clarke and D. Kozen, editors, *Proceedings of the Workshop on Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 371–382. Springer Verlag, June 1983.
- [34] F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001.
- [35] A. Pimlott and O. Kiselyov. Soutei, a logic based trust-management system. In *Proceedings of the Eighth International Symposium on Functional and Logic Programming*, 2006.
- [36] J. Reed. Hybridizing a logical framework. In *International Workshop on Hybrid Logic 2006 (HyLo 2006)*, *Electronic Notes in Computer Science*, August 2006.
- [37] U. Saranli and F. Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems. In *Proceedings of the International Conference on Robotics and Automation (ICRA '07)*, Rome, Italy, April 2007. IEEE Computer Society Press.
- [38] J. A. Vaughan, L. Jia, K. Mazurak, and S. Zdancewic. Evidence-based audit. In *Proceedings of the 21st IEEE Symposium on Computer Security Foundations*, 2008.