A Logic for Reasoning About Time-Dependent Access Control Policies

Henry DeYoung

May 20, 2008
CMU-CS-08-131

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Advisors:
Frank Pfenning
Deepak Garg

Submitted in partial fulfillment of the Senior Research Thesis Program.

This work was sponsored in part by the National Science Foundation under grant no. NSF-0716469 and the Air Force Research Laboratory under grant no. FA87500720028. The views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

Keywords: access control systems, authorization logic, proof-carrying authorization, temporal logic, hybrid logic

Abstract

Allowing access to resources, including data and hardware, without compromising their security is a fundamental challenge in computer science. Because of the number and complexity of authorization policies in access control systems, it is clear that ad hoc methods for specifying and enforcing policies cannot inspire a high degree of trust. Authorization logics have been proposed as a theoretically sound alternative.

However, for an authorization logic to be useful in practice, it should be able to model most, if not all, naturally occurring policy features. One common feature is the time-dependency of authorizations. For example, a user may only be permitted to access a given resource on workdays. Surprisingly, of the numerous proposals for access control logics, we know of no logic that incorporates time internally.

In an attempt to fill this void, this thesis develops a logic with explicit time that permits reasoning about complex, yet natural, time-dependent authorizations. The logic is then extended to account for authorizations that may be used only once. A careful study of the meta-theory of both logics is conducted, and the logics' rich expressive power is demonstrated through several examples. Finally, a proof checker for the latter logic is formalized and discussed.

Acknowledgments

First, I would like to express my gratitude to Frank Pfenning for taking time out of his very busy schedule to advise me this year, and for helping me gain valuable experience in the research process. I would also like to thank him for suggesting such an interesting and accessible thesis topic.

Second, I wish to thank Deepak Garg for being so exceptionally generous with his time and advice, and for his patience while introducing me to new topics and reviewing old ones.

Finally, I would like to thank my family for their love, support, and encouragement over not only the course of this thesis, but also at every step in my education.

Contents

List of Figures

Chapter 1

Introduction

The tension between protecting an object and allowing it to be used or displayed is a fundamental one, even for objects that are not digital. For example, how can intruders be prevented from reading a classified document while still allowing the members of that document's security compartment to read and edit it? Or, how can the public be prevented from using a departmental photocopier, while still allowing members of the department to use it?

Because of this fundamental tension, organizations usually establish *policies* that delineate the conditions under which an object can be accessed. These policies, along with a mechanism for their enforcement, constitute an access control system. But an access control system is valuable only if it can be trusted to be correct: the policies must allow only what is desired by the organization and the system must correctly enforce all of the policies.

As access control systems become more widespread and more complex, it is increasingly clear that ad hoc methods can no longer guarantee a sufficient level of trust in the system's correctness: a formal approach to access control is needed. One promising avenue is the use of logic for specifying policies. Given an appropriately defined logic, policies can be encoded as concrete logical structures, rather than relying on abstract policy descriptions.

But why is logic a solid foundation for access control? The specification of policies in a logic provides three important benefits. First, once written in a formal logic, policies have precisely specified meanings. The ambiguity inherent in a natural language formulation no longer exists. Instead, the semantics of the logic define the meaning of a policy exactly.

Second, by expressing them in a logic, access control policies can be enforced by proof-carrying authorization (PCA) [6, 7]. In a PCA-based access control system, each resource is guarded by a resource monitor. A user requesting access to a resource must present the corresponding resource monitor with a formal proof of why she is authorized, under the system's policies, to access that resource. The monitor then checks this proof for correctness. If the proof is correct, access is granted; if the proof is incorrect, access is denied.

In PCA, then, the logical model of access control coincides with access control in the real world: access is granted in practice if, and only if, it is granted formally by the logical forms of the policies. In this way, a PCA-based implementation of an access control system is guaranteed to correctly adhere to that system's policies, whatever they may be.

Third, policies written in a logic can be subjected to extensive meta-analysis. For example, non-interference properties of the logic can be proven and used in this analysis, as demonstrated by Garg and Pfenning [24]. Potentially unintended consequences of the policies can then be discovered by automated, or semi-automated, policy analysis tools based on these properties. This and other meta-analyses increase confidence in policies' correctness.

To take advantage of these benefits, it is crucial that the underlying authorization logic be able to model as many policy motifs as possible. In some cases, if the logic cannot express a critical feature of some policy, that feature could be enforced by extra-logical methods. But by abandoning the use of logic and reverting to ad hoc methods, the above benefits will no longer apply to that feature. Specifically, although a PCA proof may be correct according to the logic's rules, access may still be denied due to the failure of the extra-logical checks. This destroys the correspondence between the logical model of access and access in practice. Even worse, meta-analysis of the formal policies cannot be used to *guarantee* their correctness with respect to an informal specification because the logic does not model a critical feature.

For this reason, when designing an authorization logic, common policy motifs should be considered for inclusion. One such motif is time. It is often desirable to limit the times during which a resource can be accessed or to grant authorizations that expire. For example, students should not be able to view the solutions to a homework assignment until after the due date. Because of the ubiquity of such time-dependent access control policies, one would hope that an authorization logic incorporating time exists.

Surprisingly, of the numerous logics [2, 4, 24, 23, 30, 6, 14, 29, 22] and languages [10, 18, 39] proposed in the access control literature, few allow time-dependent policies. Those that do handle time, such as SecPAL [10], do so using extra-logical mechanisms: we know of no authorization logic that incorporates time internally. This void motivates us to develop an authorization logic with time.

Because time-dependent authorizations typically use explicit times, such as "between 9am and 5pm" or "during the month of May 2008," the logic developed in this thesis incorporates explicit time intervals rather than relative times, such as "at some time in the future." For this reason, the logic is dubbed η logic, where η (spelled "eta") stands for *E*xplicit *T*ime *A*uthorization.

η logic borrows ideas from constructive hybrid logic [13, 37, 16, 11] to model time intervals as possible worlds in which propositions may be true. Accordingly, the @ connective of hybrid logic is used to relativize the truth of a proposition to a time interval, as in $A \mathbin{@} I$. η logic also adopts techniques from constraint-based reasoning [38, 27] to manage an inclusion relation between intervals.

Another common policy motif is that of consumable credentials. One often wants to allow only a finite number of accesses. For example, students might be freely authorized to make 250 photocopies per semester and must purchase the authorization to make additional copies. That is, a finite number of accesses are free of charge.

An authorization logic that can express changes of state would be able to account for such policies. Linear logic [26, 17] is a logic that can model consumable resources. For this reason, logics of authorization that include ideas from linear logic have been proposed [23, 14]. To incorporate linear policies in addition to time-dependent ones, η logic is extended with techniques from linear authorization logics. Thus, η logic is actually a family of logics comprised of a non-linear η logic, η_N logic, and a linear η logic, η_L logic.

In summary, this thesis makes two conceptual contributions. First, an authorization logic that directly incorporates time is developed and its applicability to natural time-dependent policies is demonstrated. Second, the linear version of η logic shows that linearity can peacefully coexist with explicit time. This was not initially obvious because both linearity and time "consume" objects: linearity by usage and time by expiration.

This thesis also makes a small practical contribution. The natural deduction proof checker presented shows that a full-fledged PCA implementation of η_L logic should be easily constructible, at least in a centralized system.

Finally, it should be noted that this thesis lies in the same line as earlier joint work with Garg and Pfenning [19], and describes the latest version of η logic.

1.1 Related Work

Authorization Logics and Languages. We provide only a brief overview of the vast literature on logics and languages for access control. For more information, the reader is referred to a survey by Abadi [1].

The study of access control logics was initiated by Abadi, Burrows, and Lampson [4, 29] in two landmark papers. This work was the first to introduce the "says" connective for modeling the policies of principals, a connective found in nearly all authorization logics that followed. The work also considered an algebra over principals to model groups, delegation, and jointly made policies. η logic does not include such an algebra in this thesis, and instead relies on universal quantification to account for groups and limited delegation, as in earlier work [24].

Breaking from the classical pattern of the work of Abadi *et al.*, Garg and Pfenning [24] were the first to propose a *constructive* authorization logic. In addition, they proved several non-interference theorems for their logic. Garg *et al.* [23] continued this study by adding linearity and knowledge to their logic. η logic is primarily derived from these constructive authorization logics. The affirmation judgment, linearity, judgmental formulation, and constructive philosophy are all adopted from those works.

Subsequently, Abadi interpreted work on the Dependency Core Calculus [3] as a calculus of access control, obtaining a logic with lax-like modalities [2] similar to that of Garg and Pfenning. However, Abadi significantly extended the earlier work by describing a different, but related, non-interference theorem, proof terms through the Curry-Howard isomorphism, and second order quantification.

Notable policy languages for access control include SecPAL [10] and Binder [18]. SecPAL is the only authorization logic or language that we know of to handle time. However, in SecPAL, time restrictions are enforced by an external constraint mechanism and not reasoned about within the language. Binder extends the datalog logic programming language with constructs for reasoning about authorization.

Logics for Time. Incorporating time into logics has been the subject of much study. The most common class is that of temporal logic [31] in which times are relative to each other. Most temporal logics include the $\Box$, $\Diamond$, and $\bigcirc$ modalities for representing all future times, some future time, and the next time (in a discrete system), relative to the current time. But, having only these relative modalities, temporal logics cannot refer to absolute times. Temporal logics were indeed briefly considered for the design of a time-dependent authorization logic in this thesis. However, because access control policies typically refer to absolute, and not relative, times, this approach was rejected.

Closely related to traditional temporal logic is interval temporal logic [33], which represents an interval as a discrete sequence of events. Again, these times are relative and seem unsuitable for authorization policies.

In a departure from the temporal logic paradigm, Kanovich *et al.* [28] have modeled real-time systems using linear logic and a distinguished predicate $\mathsf{Time}(t)$ to represent the time t. While this approach permits the absolute representation of time that appears necessary for access control policies, it still seems too weak in a different way. For example, it is not clear how to express a conjunction of A occurring at time t_A and B occurring at time t_B.

Closest in spirit to η logic is Temporal Annotated Constraint Logic Programming (TACLP) [21]. TACLP contains a connective similar to the @ connective of η logic, but only allows it to annotate *atomic* propositions with time intervals. TACLP also differs from η logic in that the former is not an authorization logic; the fundamental interaction between time and authorization handled by η logic is nontrivial, and is therefore a unique contribution of η logic.

Hybrid Logic. To represent the interval at which a proposition is true, η logic borrows ideas from hybrid logic [11]. Hybrid logic extends modal logic by allowing the possible worlds to appear within propositions.

As η logic is constructive, it is most closely related to the constructive hybrid logics presented by Bräuner and de Paiva [13] and Chadha *et al.* [16]. Reed [37] also describes a constructive hybrid logical framework that inspired the hybrid features of η logic.

Constraint-Based Reasoning. The incorporation of constraints into the proof theory of η logic is heavily derived from earlier work by Saranlı and Pfenning [38] on a logic for robotic planning and work by Jia [27] in the context of reasoning about memory invariants. Jia's comments on the tradeoffs of including disjunctive constraints informed the decision to keep the constraints of η logic simple.

Linear Logic. In his influential paper on the subject, Girard pioneered linear logic [26], a logic for modeling consumable resources and other kinds of mutable state. Later, Chang *et al.* [17] introduced a judgmental reconstruction of intuitionistic linear logic by refining the hypothetical judgment. As η_L logic is also based on judgmental principles, a similar refinement is used.

The inclusion of linearity in an authorization logic to model finitely-usable credentials is certainly not unique to η_L logic; it was first adopted by Garg *et al.* [23] and independently discovered by Cederquist *et al.* [14].

Enforcing the single-use nature of consumable certificates in an implementation of a linear authorization logic is relatively straightforward if the certificates are stored in a central database. Bowers *et al.* [12] discuss the more difficult problem of coordinating accurate consumption of distributed certificates, and suggest contract-signing protocols as a solution.

Proof-Carrying Authorization. Appel and Felten [6] introduced proof-carrying authorization

(PCA) as a mechanism for enforcing access control policies via higher-order logic. In their interpretation, security policies are written in an application-specific logic and encoded in higher-order logic. Then, before access will be granted, the user must construct a correct, explicit proof of why the security policies justify access. Thus, access in practice and access in the logic coincide. To take advantage of this correspondence, we intend that η logic be amenable to PCA enforcement.

Previous implementations of PCA have been used to control access to webpages [7] and offices [9, 8]. Recently, Vaughan *et al.* [39] have designed a strongly typed language that directly incorporates PCA. It would be interesting to explore how time could be added to this language by using ideas from η logic.

Organization of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 reviews a non-linear logic of authorization that does not use time. Examples are given to clarify the use of the logic and demonstrate the need for time-dependent policies. In Chapter 3, we develop η_N logic. Examples highlight the increased expressive power of η_N logic and indicate the need for linear policies. Meta-theoretic properties of the logic are proven, increasing our confidence in the logic's soundness. Chapter 4 extends the previous logic by adding linearity. As the examples show, linearity increases the expressive power even more. The meta-theoretic properties are also extended to account for linearity. Finally, Chapter 5 presents a natural deduction formulation of η_L logic and briefly describes the corresponding implementation.

Chapter 2

Preliminaries: Garg-Pfenning Authorization Logic

η logic draws very heavily from a constructive, proof-theoretic authorization logic developed by Garg and Pfenning [24]. Before presenting η logic, it will be useful to review this logic (hereafter GP logic). This review will allow us to introduce concepts from proof-theoretic authorization logics, including the key concept of affirmation, will familiarize the reader with the expression of access control policies in an authorization logic through two examples, and will afford us an opportunity to present some meta-theory.

2.1 Logical System

Proof-theoretic logics, as an alternative to axiomatic logics, were first introduced by Gentzen [25]. These logics make the meanings of propositions exact by precisely specifying how each form of proposition may be verified. By coinciding a logic's semantics with its syntactic proofs, proof theory provides a high degree of assurance in that logic's correctness.

Later, Martin-Löf introduced a distinction between judgments and propositions [32]. Under this formulation, a judgment is an object of knowledge and is made evident by a formal proof. Propositions, then, are the subjects of judgments.

GP logic adheres to both of these fundamental ideas in an effort to keep the meanings of proofs clean and direct. (For details on this judgmental approach, the reader is referred to a discussion by Pfenning and Davies [34].) We begin by reviewing the first-order terms and sorts of the logic. Next, we introduce the truth and affirmation judgments that form the foundation of GP logic. This introduction is carefully separated from the description of the logic's propositions, to emphasize Martin-Löf's distinction. Finally, we present the proof rules of GP logic as a Gentzen-style sequent calculus.

2.1.1 First-order Terms and Sorts

To account for atomic propositions built from predicates and for universal and existential quantification, GP logic contains terms t which are classified by sorts s. That term t has sort s is denoted by the judgment $t{:}s$.

The particular sorts and terms available in GP logic are left open-ended, with the exception that a sort $\mathsf{principal}$ of principals is specifically assumed. Principals are the entities, typically users or machines, that can make statements of affirmation. The meta-variable K is used to stand for an arbitrary principal.

Because we will want to be able to reason parametrically with terms, GP logic introduces a context[1], Σ, to track the parameters in scope and their respective sorts. The syntax of a parameter context is:

$$\Sigma ::= \cdot \mid \Sigma, x{:}s$$

Thus, a parameter context is simply a map of parameter-sort pairings: it may be empty, written as $\cdot$; or, it may be a parameter context Σ followed by the ascription of a sort s to a parameter x, written as $\Sigma, x{:}s$. To avoid ambiguities, we assume that all parameters declared in Σ are distinct from x; this convention can be maintained by implicitly α-renaming variables.

Since GP logic includes parameters, the judgment $t{:}s$ must be extended to account for parameters, in addition to ground terms. The new judgment is:

$$\Sigma \vdash t{:}s$$

meaning that term t has sort s in parameter context Σ. In particular, $\Sigma, x{:}s \vdash x{:}s$ holds. Also, $[t/x]$ stands for the capture-avoiding substitution of term t for all occurrences of the free variable x. In particular, $[t/x]A$ is the proposition A with all free occurrences of x replaced by t.

2.1.2 Judgments

In GP logic, it is necessary to reason about the truth of propositions. That is, statements of the form "Proposition A is true" are objects of knowledge and the subjects of proofs. Following Martin-Löf's philosophy, GP logic therefore includes the judgment form A true, which presupposes that A is a well-formed proposition. For syntactic simplicity, the modifier true will often be dropped, so that A will implicitly stand for the judgment A true.

However, the truth of propositions is not a sufficiently expressive notion upon which to base an authorization logic. In addition to reasoning about objective truths, it is necessary to reason about principals' policies or intents. The approach taken by GP logic is to add a new judgment form K $\mathsf{affirms}$ A, meaning that "Principal K affirms that proposition A is true." A principal, then, issues a policy by affirming the truth of that policy. The affirmation K $\mathsf{affirms}$ A should not be interpreted narrowly as a *direct* statement of A by K. Instead, it may follow indirectly from other affirmations made by K. For example, in an implementation, K $\mathsf{affirms}$ A will be established either directly by a digital certificate signed by K containing A or indirectly by a logical derivation stemming from such certificates.

These judgments of truth and affirmation are the basic judgment forms of GP logic. However, they are of little use in isolation; in a logic, we need to be able to reason from hypotheses. The mechanism that GP logic uses is termed a hypothetical judgment or sequent, an extension of a basic judgment that explicitly lists the allowable assumptions.

[1]Although this meaning is distinct from its usage in the access control literature, we will continue to use this terminology, as it is common in logic and type theory.

Specifically, GP logic uses two hypothetical judgment forms:

$$\Sigma; \Gamma \Longrightarrow A \;\mathsf{true}$$
$$\Sigma; \Gamma \Longrightarrow K \;\mathsf{affirms}\; A$$

where Σ is a context of the parameters, ascribed with sorts, that may appear free in Γ, K, and A; and Γ is an unordered set of hypotheses of the form A true. In the following sections, we write γ in place of the basic judgment appearing to the right of $\Longrightarrow$ when its form does not matter; γ may stand for either A true or K $\mathsf{affirms}$ A.

The first of the above hypothetical judgments may be read "Under the hypotheses of Γ, proposition A is true, parametrically in the terms of Σ." Similarly, the second hypothetical judgment states "Under the hypotheses of Γ, principal K affirms that proposition A is true, parametrically in the terms of Σ."

2.1.3 Propositions

The syntax of propositions in GP logic is:

$$A, B ::= P \mid A \wedge B \mid \top \mid A \vee B \mid A \supset B \mid \forall x{:}s.A \mid \exists x{:}s.A \mid \langle K \rangle A$$

GP logic contains nearly all of the ordinary connectives from first-order logic: atomic propositions, P; conjunction, $A \wedge B$; truth, $\top$; implication, $A \supset B$; universal quantification, $\forall x{:}s.A$; and existential quantification, $\exists x{:}s.A$. But falsehood, $\perp$, is conspicuously absent. Falsehood is omitted from η_N logic for practical reasons that will be discussed in Section 3.1.4, and, for consistency, it is also omitted here. However, it should be noted that adding falsehood does not affect the logic itself; the meta-theorems presented in Section 2.3 have been verified with falsehood included.

Despite the close similarity of these propositions to those of first-order logic, there is one form of proposition that is unique to authorization logics: $\langle K \rangle A$, read "K says A". This proposition internalizes the affirmation judgment K $\mathsf{affirms}$ A, meaning that it is semantically equivalent to K $\mathsf{affirms}$ A, but is a proposition rather than a judgment.

Having an affirmation *proposition* allows affirmations to be combined with logical connectives, such as implication. For example, we could not combine the judgment K $\mathsf{affirms}$ A with the proposition B via implication because this would violate Martin-Löf's distinction between judgments and propositions: only propositions, and not judgments, can be operated on by the logical connectives. But we can combine the *proposition* $\langle K \rangle A$ with the proposition B via implication as $(\langle K \rangle A) \supset B$.

2.1.4 Inference Rules

GP logic possesses a proof-theoretic semantics, and its proof rules are thus critically important. They, and not any other external semantics, establish the meaning of the truth and affirmation judgments. We therefore proceed to present the proof rules of GP logic.

Each inference rule is written in the form:

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}\ \mathit{label}$$

This notation means that if the premise judgments $J_1, J_2, \ldots, J_n$ are evident, then the conclusion judgment J is also evident by the rule named *label*. Note that n may be 0. In this case, the rule has the form:

$$\frac{}{J}\ \mathit{label}$$

and the conclusion judgment J is always evident: such rules are axioms.

With the notation explained, we can now describe the inference rules of GP logic. We begin by examining the meaning of hypotheses through the init rule.

$$\frac{}{\Sigma; \Gamma, P \Longrightarrow P}\ \mathsf{init}$$

We would expect that an assumption A true could be used to immediately conclude A true. This is, in fact, the case. However, for technical reasons relating to proof search, we do not adopt this in its full generality as an inference rule, but instead use the above init rule which restricts the direct use of hypotheses to atomic propositions P. We can recover the more general form as a meta-theorem (Theorem 2.1, Section 2.3).

Next, we consider the rules for the affirmation judgment and its internalization as a proposition.

$$\frac{\Sigma; \Gamma \Longrightarrow A}{\Sigma; \Gamma \Longrightarrow K\ \mathsf{affirms}\ A}\ \mathsf{affirms}$$

When is an affirmation judgment evident? That is, when can we conclude that a principal K affirms the truth of proposition A? If A is true, it is made evident by a proof. When this proof is presented to K, K is confronted with irrefutable evidence of the truth of A. K cannot possibly deny the truth of A, for doing so would violate K's rationality. Instead, K must affirm it. Thus, one way of establishing K affirms A is to establish A true. This is captured by the above affirms rule.

In a sequent calculus, the meaning of each logical connective $\star$ is defined by a set of right rules and a set of left rules. Right rules show how $A \star B$ true may be established, and left rules show how a hypothesis $A \star B$ true may be used. For the $\langle\rangle$ connective, there is one right rule and one left rule:

$$\frac{\Sigma; \Gamma \Longrightarrow K\ \mathsf{affirms}\ A}{\Sigma; \Gamma \Longrightarrow \langle K \rangle A}\ \langle\rangle R \qquad\qquad \frac{\Sigma; \Gamma, \langle K \rangle A, A \Longrightarrow K\ \mathsf{affirms}\ B}{\Sigma; \Gamma, \langle K \rangle A \Longrightarrow K\ \mathsf{affirms}\ B}\ \langle\rangle L$$

The right rule, $\langle\rangle R$, specifies that $\langle K \rangle A$ true may be established by evidence that K affirms A holds. This is consistent with our above claim that the proposition $\langle K \rangle A$ is the internalization of the judgment K affirms A. We now know how to verify $\langle K \rangle A$ true, but how does one use the hypothesis $\langle K \rangle A$ true?

The left rule, $\langle\rangle L$, gives instructions for how the hypothesis $\langle K \rangle A$ true may be used. Because $\langle K \rangle A$ true represents the knowledge that K affirms A true, from K's perspective, A may as well be true. So, provided that we are reasoning about an affirmation made by K, that is, provided that we are inside K's mind, the hypotheses $\langle K \rangle A$ true and A true are equivalent.

This rule also holds principals accountable for their statements. Having affirmed A true, principal K cannot refute it, and so $\langle K \rangle A$ may be used as A true when reasoning about an affirmation made by K.

We now review the inference rules for implication and universal quantification. A reader familiar with the sequent calculus presentation of first-order logic may skip this discussion; there is nothing unique to GP logic in the remaining rules.

First, we give the rules for implication.

$$\frac{\Sigma; \Gamma, A \Longrightarrow B}{\Sigma; \Gamma \Longrightarrow A \supset B} \supset R \qquad \frac{\Sigma; \Gamma, A \supset B \Longrightarrow A \quad \Sigma; \Gamma, A \supset B, B \Longrightarrow \gamma}{\Sigma; \Gamma, A \supset B \Longrightarrow \gamma} \supset L$$

The implication $A \supset B$ may be intuitively thought of as a plan for converting a proof of A true to a proof of B true. Such a conversion can be established by assuming that a proof of A true is given and constructing a proof of B true from this assumption. This is captured by the right rule, $\supset R$. The conversion intuition also suggests that the hypothesis $A \supset B$ true can be used by executing this plan. Given A true, the plan $A \supset B$ true can be carried out to produce B true. This intuition is formalized in the left rule, $\supset L$.

Next, we give the rules for universal quantification.

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

The right rule, $\forall R$, states that $\forall x{:}s.A$ true may be verified by establishing A true for all possible terms of sort s. This is done by introducing a new parameter x of sort s and establishing A true parametrically in x. Just as the implication $A \supset B$ can be thought of as a plan for converting a proof of A true to a proof of B true, the right rule, $\forall R$, suggests that $\forall x{:}s.A$ can be thought of as a plan for creating a proof of $[t/x]A$ true for any term t of sort s. So, assuming such a plan and given a term t of sort s, the plan can be carried out to produce $[t/x]A$ true. This intuition is captured by the left rule, $\forall L$.

The remaining connectives of GP logic and their rules are taken directly from first-order logic, as for implication and universal quantification. A summary of all of the inference rules in GP logic is given Figure 2.1.

Before concluding this section, we illustrate some properties of GP logic. We write $\Longrightarrow A$ if, for all Σ, $\Sigma; \cdot \Longrightarrow A$ true is derivable, and write $\not\Longrightarrow A$ otherwise. Also, $A \equiv B$ abbreviates $(A \supset B) \wedge (B \supset A)$.

1. $\Longrightarrow A \supset (\langle K \rangle A)$
2. $\Longrightarrow (\langle K \rangle \langle K \rangle A) \supset (\langle K \rangle A)$
3. $\Longrightarrow (\langle K \rangle (A \supset B)) \supset ((\langle K \rangle A) \supset (\langle K \rangle B))$
4. $\not\Longrightarrow (\langle K \rangle A) \supset A$
5. $\not\Longrightarrow A$

Properties 1–3 show that $\langle K \rangle$ is similar to a lax modality [20]. Property 4 highlights the difference between truth and affirmation: truth is always affirmed (as shown in Property 1), but an affirmation by some principal does not entail truth. Finally, property 5 establishes the consistency of GP logic by demonstrating that an arbitrary proposition is not true *a priori*.

$\boxed{\text{Initial Rule}}$

$$\frac{}{\Sigma;\Gamma,P \Longrightarrow P}\ \mathsf{init}$$

$\boxed{\text{Affirmation and } \langle K\rangle A}$

$$\frac{\Sigma;\Gamma \Longrightarrow A}{\Sigma;\Gamma \Longrightarrow K\ \mathsf{affirms}\ A}\ \mathsf{affirms}$$

$$\frac{\Sigma;\Gamma \Longrightarrow K\ \mathsf{affirms}\ A}{\Sigma;\Gamma \Longrightarrow \langle K\rangle A}\ \langle\rangle R \qquad \frac{\Sigma;\Gamma,\langle K\rangle A, A \Longrightarrow K\ \mathsf{affirms}\ B}{\Sigma;\Gamma,\langle K\rangle A \Longrightarrow K\ \mathsf{affirms}\ B}\ \langle\rangle L$$

$\boxed{\text{Other Connectives}}$

$$\frac{\Sigma;\Gamma \Longrightarrow A \quad \Sigma;\Gamma \Longrightarrow B}{\Sigma;\Gamma \Longrightarrow A \wedge B}\ \wedge R$$

$$\frac{\Sigma;\Gamma,A\wedge B,A \Longrightarrow \gamma}{\Sigma;\Gamma,A\wedge B \Longrightarrow \gamma}\ \wedge L_1 \qquad \frac{\Sigma;\Gamma,A\wedge B,B \Longrightarrow \gamma}{\Sigma;\Gamma,A\wedge B \Longrightarrow \gamma}\ \wedge L_2$$

$$\frac{}{\Sigma;\Gamma \Longrightarrow \top}\ \top R$$

$$\frac{\Sigma;\Gamma \Longrightarrow A}{\Sigma;\Gamma \Longrightarrow A \vee B}\ \vee R_1 \qquad \frac{\Sigma;\Gamma \Longrightarrow B}{\Sigma;\Gamma \Longrightarrow A \vee B}\ \vee R_2$$

$$\frac{\Sigma;\Gamma,A\vee B,A \Longrightarrow \gamma \quad \Sigma;\Gamma,A\vee B,B \Longrightarrow \gamma}{\Sigma;\Gamma,A\vee B \Longrightarrow \gamma}\ \vee L$$

$$\frac{\Sigma;\Gamma,A \Longrightarrow B}{\Sigma;\Gamma \Longrightarrow A \supset B}\ \supset R \qquad \frac{\Sigma;\Gamma,A\supset B \Longrightarrow A \quad \Sigma;\Gamma,A\supset B,B \Longrightarrow \gamma}{\Sigma;\Gamma,A\supset B \Longrightarrow \gamma}\ \supset L$$

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

$$\frac{\Sigma \vdash t{:}s \quad \Sigma;\Gamma \Longrightarrow [t/x]A}{\Sigma;\Gamma \Longrightarrow \exists x{:}s.A}\ \exists R \qquad \frac{\Sigma,x{:}s;\Gamma,\exists x{:}s.A,A \Longrightarrow \gamma}{\Sigma;\Gamma,\exists x{:}s.A \Longrightarrow \gamma}\ \exists L$$

Figure 2.1: The inference rules for Garg-Pfenning logic.

2.2 Examples

Now that we have presented the judgments and crucial inference rules of GP logic, the reader should be sufficiently prepared to consider a few examples of policies written in GP logic. First, we present an example that will recur throughout the remainder of this thesis: controlling access to academic offices. Although this example is relatively small, it will still demonstrate the use of affirmation in GP logic, and, in later chapters, highlight the increased expressive power of η logic. Second, we examine the application of GP logic to chemical laboratory inspections.

In these examples, we adopt the conventions that $\vee$ and $\supset$ are right associative, and that binding precedence decreases in the order: $\langle\rangle$, $\vee$, $\supset$, $\forall$.

2.2.1 Office Entry

In this example, we describe two hypothetical policies for the Grey system [9, 8], an architecture for controlling entry to academic offices that was developed and is currently deployed at Carnegie Mellon University. In the Grey system, each office door is equipped with a processor that controls access to the office through PCA. Following the standard PCA methodology, the office door will unlock only if the principal requesting access presents the doorfront processor with a correct proof that, under the security policies of the system, she is authorized to enter.

For this example, we postulate the existence of an administrating principal, admin, that controls entry to the various faculty, staff, and student offices in his administrative domain. For simplicity, we also assume that the ownership relation between principals and offices is an injective function, so that each office can be named according to its owner.

Only one predicate is used here: may_enter. $\mathsf{may\_enter}(K_2, K_1)$ means that principal K_2 is allowed to enter K_1's office.

One reasonable policy to include in such a system is the authorization of every principal to enter her own office. Because admin controls each office, this policy is expressed in GP logic as:

$$\mathsf{own} : \langle \mathrm{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))\ \mathsf{true}$$

This policy may be read as "The administrator says that each principal K may enter her own office." Although extremely simple, this policy exhibits an important point. Because the certificate corresponding to an affirmation must be an independent object, it cannot contain free variables. Thus, any quantifiers must appear inside the top-level affirmation, as seen in the own policy. For example, the following logically equivalent formulation is difficult to enforce, as it requires one certificate from admin for each member of the potentially expandable set of principals:

$$\forall K{:}\mathsf{principal}.\langle \mathrm{admin} \rangle \mathsf{may\_enter}(K, K)\ \mathsf{true}$$

Another reasonable feature to have in an office access control system is the ability of each office owner to decide who may enter her office. To accomplish this, the administrator can agree to trust office owners' access control decisions:

$$\begin{aligned}\mathsf{trust} : \langle \mathrm{admin} \rangle (\forall K_1{:}\mathsf{principal}.\forall K_2{:}\mathsf{principal}.&\\ \langle K_1 \rangle \mathsf{may\_enter}(K_2, K_1) \supset&\\ \mathsf{may\_enter}(K_2, K_1))\ \mathsf{true}&\end{aligned}$$

This policy may be read as "The administrator says that, for all pairs of principals K_1 and K_2, if K_1 says K_2 may enter K_1's office, then K_2 may indeed enter K_1's office." The trust policy expresses a kind of delegation: K_1 now speaks for admin on matters of K_1's office.

To clarify how the trust policy can be used, consider a professor Alice and her graduate student Bob. Suppose that Alice is out of the office on May 7, 2008. Bob needs to retrieve a paper from Alice's office that he and Alice are collaborating on. He calls Alice and she agrees to issue the following credential:

$$\mathcal{C} : \langle \text{Alice} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})\ \mathsf{true}$$

Bob then approaches Alice's door and requests entry to her office using his cell phone. Before the door will unlock, Bob must submit a correct proof of

$$\Sigma_{\text{A,B}}; \mathsf{own}, \mathsf{trust}, \mathcal{C} \Longrightarrow \langle \text{admin} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})\ \mathsf{true}$$

where $\Sigma_{\text{A,B}}$ assigns sort $\mathsf{principal}$ to all principals in the system. That is, Bob must prove that the administrator allows him to enter Alice's office. Bob's phone constructs the required proof by simply applying the trust hypothesis to the $\mathcal{C}$ hypothesis (up to an approximation). The doorfront processor checks this proof, and, since it is correct, unlocks the door.

Although this policy serves its purpose, it is a rather coarse approximation of the behavior desired in general. It is likely that Alice wants the credential $\mathcal{C}$ to allow Bob access to her office *only* on May 7, 2008. If he needs access at a later time, he should be required to contact Alice again. But, under GP logic, once Alice issues credential $\mathcal{C}$, Bob will be able to enter her office at any time, even months or years after May 7, 2008!

As noted previously, time might be handled in such a system using extra-logical checks. But then, the proof does not accurately reflect the true state of the system: access might be denied even though the proof is correct. This inaccuracy, even for such a simple example as office entry, motivates the development of η logic. We will revisit this example in Section 3.2.1 and show that, in η logic, users can restrict access to their offices by time.

2.2.2 Chemical Laboratory Inspections

We now consider a more complicated example. Inspection duties of the United States Occupational Safety and Health Administration (OSHA) include the oversight of chemical laboratories. As a rough approximation, the inspection process can be thought of as a verification that all employees of the laboratory are "safe" in some appropriately defined way. OSHA will certify the laboratory only if this safety can be guaranteed.

To model the inspection process in GP logic, we assume the existence of a sort, lab, of chemical laboratories and the existence of a distinguished principal OSHA. The following predicates are required:

$\mathsf{is\_employee}(K, L)$	Principal K is an employee of lab L.
$\mathsf{is\_manager}(K, L)$	Principal K is a manager of lab L.
$\mathsf{is\_technician}(K, L)$	Principal K is a technician of lab L.
$\mathsf{is\_janitor}(K, L)$	Principal K is a janitor of lab L.
$\mathsf{is\_safe}(K, L)$	Principal K is safe in lab L.
$\mathsf{is\_certified}(L)$	Lab L is certified and may continue operating.

It is reasonable to assume that OSHA classifies each employee of a laboratory according to his job description. We assume that the three classes established by OSHA are: manager, technician, and janitor. This classification policy can be expressed as:

$$\begin{array}{l}\mathsf{job} : \langle \mathrm{OSHA} \rangle (\forall L{:}\mathsf{lab}.\forall K{:}\mathsf{principal}.\\ \quad \mathsf{is\_employee}(K, L) \supset\\ \quad\quad (\mathsf{is\_manager}(K, L) \vee\\ \quad\quad\; \mathsf{is\_technician}(K, L) \vee\\ \quad\quad\; \mathsf{is\_janitor}(K, L))) \ \mathsf{true}\end{array}$$

This policy provides a method for distinguishing the job that an employee holds. Employees holding different positions may be "safe" under different conditions. For example, janitors may be exposed to chemicals but need not operate lab equipment, while technicians will handle chemicals and operate equipment. For this reason, a janitor might be "safe" if he can access safety procedures for all chemicals in the lab, but he need not (and perhaps should not) access equipment manuals. On the other hand, a technician would need to be able to access both chemical safety procedures and equipment manuals to be "safe."

OSHA's certification policy can then be expressed as:

$$\begin{array}{l}\mathsf{certify} : \langle \mathrm{OSHA} \rangle (\forall L{:}\mathsf{lab}.\\ \quad (\forall K{:}\mathsf{principal}.\mathsf{is\_employee}(K, L) \supset \mathsf{is\_safe}(K, L)) \supset\\ \quad\; \mathsf{is\_certified}(L)) \ \mathsf{true}\end{array}$$

This can be read as "OSHA says that a lab L is certified if, for all employees K of lab L, K is safe in lab L."

In many policies, credentials are required to establish a result. Note that, in the certify policy, however, the requirement is a kind of conditional credential: the safety of a principal K in lab L is only needed when K is an employee of L. Because this condition exists, it is possible, using the case analysis induced by the job policy, to take the specific job of K into account when determining K's safety.

2.3 Meta-theory

One of the key advantages of a proof-theoretic logic is its singular amenability to a rigorous meta-theoretic analysis. Meta-theorems are stated as natural and desirable properties of the logic—properties that one would expect to hold. The proofs of these properties serve as a "sanity check" on the design of the logic; if some expected property fails to hold, perhaps the logic's design should be reconsidered.

As a proof-theoretic logic, the meta-theory of GP logic can be explored in this way. There are two reasonable properties for GP logic. First, as alluded to in the discussion of the init rule (cf. Section 2.1.4), for *any* proposition A, from the assumption that A is true, it should be possible to establish that A is true. For atomic propositions P, this is captured explicitly in the init inference rule. For arbitrary propositions A, this is stated and proved as the following theorem.

Theorem 2.1 (Identity)**.** *For any proposition A, $\Sigma; \Gamma, A \ \mathsf{true} \Longrightarrow A \ \mathsf{true}$.*

Second, the logic should possess the cut elimination property. One cut rule for GP logic states that a proof of $A\ \mathsf{true}$ can be used to replace the hypothesis $A\ \mathsf{true}$ in a proof of γ to yield a direct proof of γ. For this reason, a cut rule might be intuitively thought of as a method for creating and using lemmata: the proof of $A\ \mathsf{true}$ functions as the lemma and the hypothesis $A\ \mathsf{true}$ in the proof of γ corresponds to the use of the lemma in proving the main theorem.

Since GP logic also permits conclusions of the form $K\ \mathsf{affirms}\ A$, a cut rule for affirmation is also needed. $K\ \mathsf{affirms}\ A$ can replace the hypothesis $A\ \mathsf{true}$ in a proof of $K\ \mathsf{affirms}\ B$ since, from K's perspective, truth and K's affirmations are equivalent.

Cut elimination means that an explicit cut rule is not needed in the logic: any uses of the rule are unnecessary. The following theorem states the admissibility of cut. Because cut elimination follows from this by a straightforward induction, often only the admissibility of cut is formally stated and proven.

Theorem 2.2 (Admissibility of Cut)**.**

1. *If* $\Sigma; \Gamma \Longrightarrow A\ \mathsf{true}$ *and* $\Sigma; \Gamma, A\ \mathsf{true} \Longrightarrow \gamma$*, then* $\Sigma; \Gamma \Longrightarrow \gamma$.
2. *If* $\Sigma; \Gamma \Longrightarrow K\ \mathsf{affirms}\ A$ *and* $\Sigma; \Gamma, A\ \mathsf{true} \Longrightarrow K\ \mathsf{affirms}\ B$*, then* $\Sigma; \Gamma \Longrightarrow K\ \mathsf{affirms}\ B$.

The proofs and associated lemmata for the above meta-theorems are given in [24].

2.4 Conclusion

In hopes of adequately preparing the reader for the following discussion of η logic, this chapter has reviewed a proof-theoretic authorization logic developed by Garg and Pfenning [24]. We have also seen the application of GP logic to two disparate systems: office access control and chemical laboratory inspections. Finally, we have presented the meta-theory of GP logic and explained its importance as an expression of the logic's soundness. We now proceed to develop η_N logic, which is heavily based on principles from GP logic reviewed in this chapter.

Chapter 3

η_N Logic

As reviewed in the preceding chapter, an authorization logic can form a theoretically sound basis for access control systems. However, as demonstrated in the office entry example, it is necessary that the logic express time-dependent policies in order to facilitate more accurate models of access control in practice.

In this chapter, we develop an authorization logic with explicit time, η_N logic, by modifying GP logic. After giving a formal description of η_N logic, we present two applications of the logic to systems with time-dependent policies. Finally, we carry out a careful study of the logic's meta-theory and establish a formal correspondence with GP logic.

3.1 Logical System

η_N logic synthesizes ideas from several diverse logics. The concept of affirmation is borrowed from GP logic, the notion of truth relativized to an interval is inspired by the use of worlds in hybrid logic, and the combination of constraints and proof theory is adapted from constraint-based logics.

The presentation of the logic is therefore broken into several sections. We begin by discussing first-order terms and sorts, followed by a description of the system of constraints. Next, we introduce the judgments and propositions of the logic. Finally, we construct a proof-theoretic semantics for the logic by giving the inference rules.

3.1.1 First-order Terms and Sorts

The basic system for first-order terms and sorts remains as it is in GP logic. Σ is still a context listing the parameters in scope and their respective sorts. We continue to write $\Sigma \vdash t{:}s$ for the judgment that term t has sort s and $[t/x]A$ for the substitution of term t for the free variable x in proposition A.

The sort $\mathsf{principal}$ of principals is carried over from GP logic. η_N logic includes two additional sorts: the sort time of times and the sort $\mathsf{interval}$ of time intervals. Application-specific sorts can be added as needed.

Times are the components that comprise the time intervals about which η_N logic reasons. Because the logic does not depend on it, a concrete structure for times is not given, but instead left to be specified by individual applications. However, one may intuitively think of times as points on the real line. Times are usually represented by t; it should be clear from the context whether a given occurrence of t indicates an arbitrary term or a time.

Intervals, represented with the meta-variable I, are sets of time about which reasoning occurs. Despite the use of the terminology "interval," these sets of time need not be intervals in the mathematical sense; that is, they need not have the form $[t_1, t_2] = \{x \mid t_1 \leq x \leq t_2\}$ or the related open interval forms. η_N logic is flexible enough to permit the use of arbitrary sets of time. However, we overlook the slight abuse of terminology since sets that are strictly intervals appear naturally in many applications.

3.1.2 Constraints

As will be seen in Section 3.1.5, the rules of η_N logic will require an inclusion relation for intervals. Because interval parameters are permitted in the logic, it is not sufficient to simply adopt a mathematical definition of interval inclusion. Instead, a constraint domain is incorporated in the logic. The superset constraint form $I \supseteq I'$ is required, but the remainder of this domain is left open-ended: other constraint forms may be freely added for application-specific purposes. The meta-variable C denotes an arbitrary constraint form.

Because it will be necessary to assume that certain constraints hold during reasoning, a constraint context is introduced, with the following syntax:

$$\Psi ::= \cdot \mid \Psi, C$$

Thus, each constraint context Ψ is a (possibly empty) set of constraints. Reordering of the members of Ψ is freely permitted. We will use the constraint entailment judgment

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

to mean "Under the constraints of Ψ, constraint C holds, parametrically in the members of Σ." Note that the context Σ is required because Ψ and C may contain parameters from Σ.

Because the structure of intervals is left abstract, even the particular decision procedure used to solve superset constraints remains relatively unspecified: any system satisfying the following six basic properties can be used as the constraint domain. These properties are required for the meta-theory that will be presented in Section 3.3.

(Hypothesis) $\Sigma; \Psi, C \models C$.
(Weakening) If $\Sigma; \Psi \models C$, then $\Sigma, \Sigma'; \Psi, \Psi' \models C$.
(Cut) If $\Sigma; \Psi \models C$ and $\Sigma; \Psi, C \models C'$, then $\Sigma; \Psi \models C'$.
(Substitution) If $\Sigma \vdash t{:}s$ and $\Sigma, x{:}s; \Psi \models C$, then $\Sigma; [t/x]\Psi \models [t/x]C$.
(Reflexivity) $\Sigma; \Psi \models I \supseteq I$.
(Transitivity) If $\Sigma; \Psi \models I \supseteq I'$ and $\Sigma; \Psi \models I' \supseteq I''$, then $\Sigma; \Psi \models I \supseteq I''$.

3.1.3 Judgments

Our goal in designing η_N logic is to allow reasoning about explicit time within an authorization logic. Instead of reasoning about the truth of propositions, as was done in GP logic, it is necessary to reason about the truth of propositions *during* explicit time intervals. Therefore, the objects of knowledge in η_N logic are not statements of the form "Proposition A is true," but rather "Proposition A is true during interval I." According to Martin-Löf's philosophy, the logic should therefore include a judgment form that relativizes truth to a time interval. We choose to write $A[I]$ for the judgment meaning "Proposition A is true during interval I."

In addition to its truth judgment form, A true, GP logic includes an affirmation judgment form, K $\mathsf{affirms}$ A, to model principals' intents and policies. It is therefore natural to include affirmation in η_N logic, since it is still necessary to model policies. But how should affirmation interact with explicit time intervals?

By adopting the reasonable notion that everything can be relativized to a time interval, it can be concluded that each affirmation made by a principal occurs on some time interval. Moreover, a principal cannot affirm a proposition, but must instead affirm a judgment. Combining these two ideas naturally leads to statements of the form "During interval I, principal K affirms the truth of proposition A on interval I'" as objects of knowledge. Using the @ connective described in the next two sections, the previous statement will be equivalent to "During interval I, principal K affirms the truth of proposition A @ I' on interval I." As a result, it is sufficient to consider only statements of the latter form: if the interval of truth is different than the interval of affirmation, it can be embedded in the proposition.

We therefore arrive at the judgment form $(K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I$ meaning that "During interval I, principal K affirms the truth of proposition A on I." Since, as mentioned previously, principals do not affirm propositions, but instead affirm judgments, it would be more precise to write the affirmation judgment form as $(K\ \mathsf{affirms}\ A[I])\ \mathsf{at}\ I$. But because the two intervals are the same, we can elide the first interval.

Because reasoning from assumptions is needed, η_N logic extends the basic judgment forms $A[I]$ and $(K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I$ to permit hypotheses. The hypothetical judgment forms are:

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

where Σ is a context ascribing sorts to the parameters that may appear in Ψ, Γ, K, A, and I; Ψ is a constraint context containing the constraints assumed to hold; and Γ is a set of hypotheses of the form $A[I]$. In the following sections, we will write γ in place of the basic judgment to the right of $\Longrightarrow$ when its form does not matter.

The first hypothetical judgment form means "Assuming that the constraints in Ψ hold and under the assumptions in Γ, proposition A is true during interval I, parametrically in the members of Σ." Similarly, the second hypothetical judgment form means "Assuming that the constraints in Ψ hold and under the assumptions in Γ, during interval I, principal K affirms that proposition A is true on I, parametrically in the members of Σ."

3.1.4 Propositions

The propositions in η_N logic are given by the following grammar:

$$\begin{aligned} A, B ::= {} & P \mid A \wedge B \mid \top \mid A \vee B \mid A \supset B \mid \forall x{:}s.A \mid \exists x{:}s.A \\ & \mid \langle K \rangle A \mid A \mathbin{@} I \mid C \mathbin{\dot{\supset}} A \mid C \mathbin{\dot{\wedge}} A \end{aligned}$$

These propositions include those of GP logic. Just as $\langle K\rangle A$ internalizes the judgment K $\mathsf{affirms}$ A in GP logic, $\langle K\rangle A$ internalizes the judgment $(K \mathsf{~affirms~} A) \mathsf{~at~} I$ in η_N logic. Although the formal meanings of the connectives must shift with the change from time-independent basic judgments to time-dependent ones, the connectives still retain their intuitive meanings. For example, $A \wedge B$ still behaves like a pair of A and B. This is made precise by the formal correspondence between GP logic and a fragment of η_N logic established in Section 3.3.2.

There are three new proposition forms in η_N logic: $A@I$, $C \dot{\supset} A$, and $C \dot{\wedge} A$. The proposition $A@I$ internalizes the new judgment $A[I]$, allowing us to legitimately combine it with the other logical connectives. For example, although $(A[I]) \supset B$ would violate the distinction between judgments and propositions, $(A @ I) \supset B$ is a well-formed proposition.

$C \dot{\supset} A$ and $C \dot{\wedge} A$ are constraint implication and constraint conjuction propositions, respectively, adapted from Saranlı and Pfenning's Constrained Intuitionistic Linear Logic [38]. They permit the constraint domain to interact with the rest of the logic.

It should be noted that falsehood, $\bot$, is not included in the logic, stemming from the need to avoid security risks. If falsehood was included and, by some accident of policy management, a contradiction existed for any interval I, even an arbitrarily small one, then the judgment $\bot[I]$ would be derivable. From this judgment, any user would be able to give a valid proof of any judgment, including those allowing him to access protected resources even at times outside of I.[1] We therefore exclude falsehood from the logic to prevent this scenario from ever arising.

One consequence of the absence of falsehood is that policies to explicitly deny a group of users access cannot be written; only policies that explicitly *allow* a group of users access can be written. Stated differently, only whitelists, and not blacklists, can be created.

3.1.5 Inference Rules

Following the presentation of GP logic, we now state a few key proof rules and attempt to provide some intuition for them. We postpone the inference rules for the well-formedness of propositions, judgments, and contexts to Section 5.1 to avoid obscuring the key proof rules.

We begin by presenting the init rule that defines the nature of hypotheses:

$$\frac{\Sigma; \Psi \models I \supseteq I'}{\Sigma; \Psi; \Gamma, P[I] \Longrightarrow P[I']}\ \mathsf{init}$$

We would expect that, from the assumption that proposition A is true on interval I, it should be possible to prove that A is true on I. More generally, since truth on an interval refers to truth over the whole of that interval, it should be possible to prove that A is true on any subinterval I' of I from this assumption. The init rule captures this intuition, though, as in GP logic, it is restricted to atomic propositions P for technical reasons relating to proof search. The init rule in its full generality is proven admissible in Theorem 3.2 (cf. Section 3.3.1).

Next, consider the new connective: @.

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

[1]The admissibility of cut requires that a $\bot[I]$ hypothesis prove judgments at *arbitrary* intervals, not just at I.

The right rule, $@R$, shows that establishing $A[I]$ is sufficient evidence for $A @ I[I']$, for any interval I'. The left rule, $@L$, allows the hypothesis $A @ I[I']$ to be used as $A[I]$.

Taken together, these rules imply an equivalence between $A[I]$ and $A @ I[I']$ for any I', and also show that $A @ I$ internalizes the hybrid judgment $A[I]$. For example, establishing that "In 2008, it is true that 'During 1815–1821, Napoleon Bonaparte is in exile'" is equivalent to establishing that "During 1815–1821, Napoleon Bonaparte is in exile." In other words, whether it is true *now* that Napoleon was in exile depends only on whether it was true *then* that Napoleon was in exile.

Next, we examine the constraint connectives. First, the rules for constraint implication:

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

$C \mathbin{\dot{\supset}} A$ represents the proposition A with the constraint precondition C. Thus, as formalized in the $\dot{\supset}R$ rule, verifying $C \mathbin{\dot{\supset}} A[I]$ involves verifying that A is true during interval I under the assumption that constraint C holds. The $\dot{\supset}L$ rule states that to establish $A[I]$ from $C \mathbin{\dot{\supset}} A[I]$, one must simply establish the constraint precondition C.

The other constraint connective is constraint conjunction.

$$\frac{\Sigma; \Psi \models C \quad \Sigma; \Psi; \Gamma \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma \Longrightarrow C \mathbin{\dot{\wedge}} A[I]}\ \dot{\wedge}R \qquad \frac{\Sigma; \Psi, C; \Gamma, C \mathbin{\dot{\wedge}} A[I], A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, C \mathbin{\dot{\wedge}} A[I] \Longrightarrow \gamma}\ \dot{\wedge}L$$

The $\dot{\wedge}R$ rule requires that the constraint C hold and that A be true during interval I, reminiscent of the right rule for ordinary conjunction. The $\dot{\wedge}L$ rule allows the hypothesis $C \mathbin{\dot{\wedge}} A[I]$ to be used by projecting out the two component hypotheses: C and $A[I]$.

Next, we consider the rules for the affirmation judgment $(K \,\mathsf{affirms}\, A) \,\mathsf{at}\, I$ and its internalization as $\langle K \rangle A$.

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

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

The $\mathsf{affirms}$ rule indicates that, during interval I, every principal K is prepared to affirm the truth of A on I if confronted with incontrovertible evidence of it: K cannot possibly ignore the evidence and must therefore affirm $A[I]$.

The right rule, $\langle\rangle R$, shows that $\langle K \rangle A$ internalizes the affirmation judgment $(K \ \mathsf{affirms}\ A) \ \mathsf{at}\ I$. That is, by establishing $(K \ \mathsf{affirms}\ A) \ \mathsf{at}\ I$, one may conclude that the proposition $\langle K \rangle A$ is true on interval I.

The left rule, $\langle\rangle L$, shows how to use an affirmation made by K during interval I. As in GP logic, the distinction between K's affirmations and truth disappears when trying to prove an affirmation made by K. However, with time-dependent affirmations, the disappearance of this distinction is only valid for affirmations made by K during a superinterval I of the interval I' for the affirmation made by K that is being established. Without the interval constraint, this rule would be incorrect. If I is not a superinterval of I', one cannot be assured that K still affirms A during all of interval I'.

Next, we examine implication. Implication interacts very strongly with time, as evidenced by the combination of parameters, constraints, and hybrid worlds in its right and left rules:

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

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

The judgment $A \supset B[I]$ may be intuitively thought of as a plan for converting A to B that is available during any subinterval of I. Such a conversion can be established by deriving $B[i]$ under the assumption $A[i]$, parametrically in the arbitrary subinterval i of I. The parameter and corresponding constraint ensure that the conversion is valid at every time in I. This intuition is formalized in the right rule, $\supset R$.

The conversion intuition also appears in the left rule, $\supset L$. The plan $A \supset B[I]$ for converting A to B can be carried out to produce $B[I']$ from $A[I']$, provided I' is a subinterval of I. The rule is incorrect without the subinterval proviso because the plan would not be available at an arbitrary I'.

The remaining connectives interact with time in straightforward ways. One such connective is conjunction:

$$\frac{\Sigma; \Psi; \Gamma \Longrightarrow A[I] \quad \Sigma; \Psi; \Gamma \Longrightarrow B[I]}{\Sigma; \Psi; \Gamma \Longrightarrow A \wedge B[I]} \wedge R$$

$$\frac{\Sigma; \Psi; \Gamma, A \wedge B[I], A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A \wedge B[I] \Longrightarrow \gamma} \wedge L_1 \qquad \frac{\Sigma; \Psi; \Gamma, A \wedge B[I], B[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A \wedge B[I] \Longrightarrow \gamma} \wedge L_2$$

To show that $A \wedge B$ is true on interval I, it is sufficient to show both that A is true on I and that B is true on I; this is captured by the $\wedge R$ rule. The left rules, $\wedge L_1$ and $\wedge L_2$, show that both A and B are true on I if $A \wedge B$ is true on I. These right and left rules do not manipulate the interval annotations; they are the same as the rules in first-order logic for conjunction, but are tagged with intervals.

Note that these rules and the rules for $\supset$ imply the usual equivalence of curried and uncurried implications: $A \supset (B \supset C)[I]$ and $(A \wedge B) \supset C[I]$ entail each other.

The remaining proof rules follow the pattern of the rules for conjunction, and are given in Figure 3.1.

Before concluding this section, we state some properties of η_N logic. We write $\Longrightarrow A$ if, for all Σ, Ψ, and I'', $\Sigma; \Psi; \cdot \Longrightarrow A[I'']$ is derivable, and write $\not\Longrightarrow A$ otherwise. Also, $A \equiv B$ abbreviates $(A \supset B) \wedge (B \supset A)$.

First, as in GP logic, $\langle K \rangle$ is similar to a lax modality [20] and $\langle K \rangle A$ does not imply A in general:

1. $\Longrightarrow A \supset (\langle K \rangle A)$
2. $\Longrightarrow (\langle K \rangle \langle K \rangle A) \supset (\langle K \rangle A)$
3. $\Longrightarrow (\langle K \rangle (A \supset B)) \supset ((\langle K \rangle A) \supset (\langle K \rangle B))$

$\boxed{\text{Initial Rule}}$

$$\frac{\Sigma; \Psi \models I \supseteq I'}{\Sigma; \Psi; \Gamma, P[I] \Longrightarrow P[I']}\ \mathsf{init}$$

$\boxed{A \mathbin{@} I}$

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

$\boxed{\text{Constraints}}$

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

$$\frac{\Sigma; \Psi \models C \quad \Sigma; \Psi; \Gamma \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma \Longrightarrow C \mathbin{\dot{\wedge}} A[I]}\ \dot{\wedge}R \qquad\qquad \frac{\Sigma; \Psi, C; \Gamma, C \mathbin{\dot{\wedge}} A[I], A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, C \mathbin{\dot{\wedge}} A[I] \Longrightarrow \gamma}\ \dot{\wedge}L$$

$\boxed{\text{Affirmation and } \langle K \rangle A}$

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

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

$\boxed{\text{Other Connectives}}$

$$\frac{\Sigma; \Psi; \Gamma \Longrightarrow A[I] \quad \Sigma; \Psi; \Gamma \Longrightarrow B[I]}{\Sigma; \Psi; \Gamma \Longrightarrow A \wedge B[I]}\ \wedge R$$

$$\frac{\Sigma; \Psi; \Gamma, A \wedge B[I], A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A \wedge B[I] \Longrightarrow \gamma}\ \wedge L_1 \qquad\qquad \frac{\Sigma; \Psi; \Gamma, A \wedge B[I], B[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A \wedge B[I] \Longrightarrow \gamma}\ \wedge L_2$$

$$\frac{}{\Sigma; \Psi; \Gamma \Longrightarrow \top[I]}\ \top R$$

$$\frac{\Sigma; \Psi; \Gamma \Longrightarrow A[I]}{\Sigma; \Psi; \Gamma \Longrightarrow A \vee B[I]}\ \vee R_1 \qquad\qquad \frac{\Sigma; \Psi; \Gamma \Longrightarrow B[I]}{\Sigma; \Psi; \Gamma \Longrightarrow A \vee B[I]}\ \vee R_2$$

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

Figure 3.1: The inference rules for η_N logic.

$\boxed{\text{Other Connectives, cont.}}$

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

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

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

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

Figure 3.2: The inference rules for η_N logic, continued.

4. $\not\Longrightarrow (\langle K \rangle A) \supset A$

Next, we state a few properties of the @ connective:

5. $\not\Longrightarrow (A \mathbin{@} I) \supset (A \mathbin{@} I')$
6. $\Longrightarrow (I \supseteq I') \mathbin{\dot\supset} ((A \mathbin{@} I) \supset (A \mathbin{@} I'))$
7. $\Longrightarrow (A \mathbin{@} I) \equiv (A \mathbin{@} I \mathbin{@} I')$
8. $\Longrightarrow ((A \wedge B) \mathbin{@} I) \equiv ((A \mathbin{@} I) \wedge (B \mathbin{@} I))$
9. $\Longrightarrow ((A \vee B) \mathbin{@} I) \equiv ((A \mathbin{@} I) \vee (B \mathbin{@} I))$
10. $\Longrightarrow ((A \supset B) \mathbin{@} I) \supset ((A \mathbin{@} I) \supset (B \mathbin{@} I))$
11. $\not\Longrightarrow ((A \mathbin{@} I) \supset (B \mathbin{@} I)) \supset ((A \supset B) \mathbin{@} I)$
12. $\not\Longrightarrow (((\langle K \rangle A) \mathbin{@} I) \supset (\langle K \rangle (A \mathbin{@} I))$
13. $\not\Longrightarrow (\langle K \rangle (A \mathbin{@} I)) \supset (((\langle K \rangle A) \mathbin{@} I)$

Property 5 shows that truth on one interval does not entail truth on another interval, in general; the intervals may be unrelated. When the intervals are related by inclusion, the entailment does hold, as in property 6. Property 7 indicates that only the innermost $@\, I$ matters. This relates to the previously mentioned equivalence between $A[I]$ and $A \mathbin{@} I[I']$. Properties 8 and 9 show that @ naturally distributes over $\wedge$ and $\vee$, intuitive and desirable properties. Properties 10 and 11 demonstrate that @ distributes over $\supset$ only in one direction. Properties 12 and 13 state that @ does not commute with $\langle K \rangle$.

Finally, η_N logic is consistent:

14. $\not\Longrightarrow A$

3.2 Examples

With η_N logic formalized, we can now illustrate its application to time-dependent access control policies by considering two examples. First, we revisit the office entry example and refine its policies to incorporate time. Second, we examine a journal publication system.

In these examples, we adopt the conventions that $\supset$ and $\dot{\supset}$ are right associative, and that binding precedence decreases in the order: $\langle\rangle$; @; $\supset$ and $\dot{\supset}$; $\forall$.

3.2.1 Office Entry

Recall, from Section 2.2.1, the office entry example that was based on the Grey system. This example assumed an administrating principal, admin, that controlled entry to the offices, named each office according to its owner, and used the predicate may_enter, where $\mathsf{may\_enter}(K_2, K_1)$ meant that K_2 may enter K_1's office. The two policies proposed for a PCA architecture based on GP logic were:

$$\mathsf{own} : \langle \text{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))\ \mathsf{true}$$

$$\begin{aligned}\mathsf{trust} : \langle \text{admin} \rangle (\forall K_1{:}\mathsf{principal}.\forall K_2{:}\mathsf{principal}.\\ \langle K_1 \rangle \mathsf{may\_enter}(K_2, K_1) \supset\\ \mathsf{may\_enter}(K_2, K_1))\ \mathsf{true}\end{aligned}$$

The first of these policies allowed every office owner to enter her own office. The second policy allowed an office owner to make decisions about who may enter her office, decisions which the administrator trusted.

The above GP logic policies were sufficient for controlling who could enter an office, but not for controlling *when* that person could enter. This deficiency resulted from the inability of GP logic to reason with time internally. Now that we have developed η_N logic as an authorization logic with time, it is natural to check that the new logic is expressive enough to handle time-based office entry policies.

First, consider creating a η_N logic analogue of the own policy. Because η_N logic includes all connectives from GP logic and because these connectives retain their intuitive meanings, a natural attempt uses the same proposition as own:

$$\langle \text{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))[?]$$

At this moment, the judgment is incomplete: the time interval over which the proposition is true has not yet been specified (indicated by '?').

What interval should be used? It must be the same as the interval over which the policy will be valid. If the administrator wants to allow each office owner to enter her own office only during interval I, then the interval for this policy should be I. In the setting of academic offices, it would seem unusual for an office owner to be prevented from entering her office at any time. So, in this specific instance, the interval is $(-\infty, \infty)$. The η_N logic analogue of own is then:

$$\mathsf{own}' : \langle \text{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))[(-\infty, \infty)]$$

This policy means that "At all times, the administrator says that each principal K may enter her own office at any time."

Note that the administrator need not commit to a policy for an extended period of time. For example, suppose that the administrator only wants to commit to allowing an office owner to enter her own office during 2008. The administrator would issue the policy with 2008 as its validity interval. If the administrator later chooses to extend the policy through 2009, he can reissue the same policy with the new interval 2009. If, instead, the administrator chooses not to renew the policy, he simply does nothing: the 2008 version will no longer be valid in 2009.

Next, consider creating an analogue of the trust policy. Again, we use the same proposition as in trust. For concreteness, we choose $(-\infty, \infty)$ as the validity interval, but it should be noted that any desired interval could be used. The policy is then:

$$\begin{aligned}\mathsf{trust}' : \langle \text{admin} \rangle (\forall K_1{:}\mathsf{principal}.\forall K_2{:}\mathsf{principal}.\\ \langle K_1 \rangle \mathsf{may\_enter}(K_2, K_1) \supset\\ \mathsf{may\_enter}(K_2, K_1))[(-\infty, \infty)]\end{aligned}$$

This policy means that "At all times, the administrator says that, for all pairs of principals K_1 and K_2, if K_1 says K_2 may enter K_1's office at some time, then K_2 may indeed enter K_1's office at that time."

With this policy, we can now reconsider the situation of the professor Alice and her graduate student Bob. Recall that Alice is out of the office on May 7, 2008 and that Bob needs to retrieve a paper from Alice's office. Alice agrees to authorize Bob to enter her office, but only for that day. So, she issues the following credential:

$$\mathcal{C}' : \langle \text{Alice} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[5/7/08]$$

At some time t, Bob will approach Alice's office door and request access using his cell phone. Before the door will unlock, he must present a correct proof of:

$$\Sigma_{\text{A,B}}; \cdot; \mathsf{own}', \mathsf{trust}', \mathcal{C}' \Longrightarrow \langle \text{admin} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[[t, t]]$$

where $\Sigma_{\text{A,B}}$ assigns the sort principal to all principals in the system. Provided that t is some time during May 7, 2008 (formally, $\models 5/7/08 \supseteq [t, t]$) Bob's phone can construct a correct proof by applying the trust$'$ policy to the credential $\mathcal{C}'$ that Alice supplied, and Bob will be granted access. If t is not during May 7, 2008 (formally $\not\models 5/7/08 \supseteq [t, t]$), there is no correct proof of the required judgment, and Bob will not be granted access.

As is evident from this example, η_N logic permits the expression of a richer set of policies than is possible in GP logic. However, the logic is still not sufficiently expressive. The deficiency occurs even in this small office entry example. Alice can now restrict the times during which Bob may access her office, but it is not possible to restrict the *number* of times Bob may enter. Specifically, because the credential $\mathcal{C}'$ is never consumed during use, Bob may enter the office as many times as he wants on May 7, 2008 by repeatedly using $\mathcal{C}'$.

Because η_N logic models the expiration of, but not the consumption of, credentials, the above deficiency motivates us to extend the logic with linearity, just as Garg and Pfenning cleanly added linearity to an authorization logic without time [23]. This effort toward a linear η logic that can model consumable credentials is the focus of the following chapter.

3.2.2 Journal Publication

To further demonstrate the increased expressiveness of η_N logic, consider a peer-review publication system as employed by academic journals. This example uses time in a more complex way than the previous example and also illustrates the use of time-based constraints and constraint implication.

We postulate the existence of two application-specific sorts: the sort journal of academic journals and the sort article of journal articles. To ease the notation, we also use the syntax $t \in I$ as an abbreviation for the constraint $I \supseteq [t, t]$. The following predicates are required:

$\mathsf{is\_approved}(A, K, J)$	Article A is approved by principal K for publication in journal J.
$\mathsf{is\_reviewer}(R, A, J)$	Principal R is the reviewer for article A submitted to journal J.
$\mathsf{is\_editor}(E, J)$	Principal E is an editor for journal J.
$\mathsf{is\_published}(A, J)$	Article A is published in journal J.

Journal J appoints E as an editor for term I by issuing the credential $\langle J\rangle\mathsf{is\_editor}(E, J)[I]$. One of an editor's duties is to assign reviewers to articles submitted to the journal. Editor E assigns principal R as the reviewer for article A from time t onward by issuing the credential $\langle E\rangle\mathsf{is\_reviewer}(R, A, J)[[t, \infty)]$. For simplicity, we assume that each article has at most one reviewer, justifing the reference to a reviewer of an article as *the* reviewer.

Another one of an editor's duties is to process reviews as they come back from reviewers. Editor E accomplishes this by signing the following credential:

$$\begin{aligned}\mathsf{approve} : \langle E\rangle(\forall R{:}\mathsf{principal}.\forall t_a{:}\mathsf{time}.\ & \\ & \langle R\rangle\mathsf{is\_approved}(A, R, J) \mathbin{@} [t_a, t_a] \supset \\ & \mathsf{is\_reviewer}(R, A, J) \mathbin{@} [t_a, t_a] \supset \\ & (t_a \in I_E) \mathbin{\dot{\supset}} \\ & \quad \mathsf{is\_approved}(A, E, J) \mathbin{@} [t_a, \infty))[(-\infty, \infty)]\end{aligned}$$

If principal R decides to approve article A for publication in journal J, he can submit a positive review at time t_a by issuing $\langle R\rangle\mathsf{is\_approved}(A, R, J)[[t_a, t_a]]$. Provided that editor E agrees that R is the reviewer of article A at time t_a and that $t_a \in I_E$, E will accept R's review and approve the article for publication from time t_a onward. If R is not the reviewer of article A or if $t_a \notin I_E$, then the review will not be accepted.

Note that, unlike the policies we have previously seen, approve is not a fixed policy, but rather a template. When E signs the credential, he must instantiate I_E with the interval over which he will accept reviews.

In a similar way, each journal must specify the conditions under which it accepts articles approved by editors. This is done by issuing the following credential:

$$\begin{aligned}\mathsf{publish} : \langle J\rangle(\forall E{:}\mathsf{principal}.\forall t_a{:}\mathsf{time}.\ & \\ & \langle E\rangle\mathsf{is\_approved}(A, E, J) \mathbin{@} [t_a, t_a] \supset \\ & \mathsf{is\_editor}(E, J) \mathbin{@} [t_a, t_a] \supset \\ & (t_a \in I_J) \mathbin{\dot{\supset}} \\ & \quad \mathsf{is\_published}(A, J) \mathbin{@} [t_a, \infty))[(-\infty, \infty)]\end{aligned}$$

If principal E approves article A for publication in journal J, he issues the consumable credential $\langle E\rangle\mathsf{is\_approved}(A, E, J)[[t_a, t_a]]$. If journal J has appointed E as editor during time t_a and if $t_a \in I_J$,

J will accept editor E's approval and publish the article from t_a onward. Again, this policy is a template: J must instantiate I_J with the interval during which it will accept articles for publication.

3.3 Meta-theory and Correspondence to GP Logic

As a proof-theoretic logic, η_N logic permits a rigorous study of its meta-theory. We examine three properties here: identity, subsumption, and admissibility of cut. In addition, we easily establish a formal correspondence between a fragment of η_N logic and GP logic, which is not surprising given the parentage relationship between the two logics.

3.3.1 Meta-theory

The meta-theory for η_N logic is slightly more complicated than that of GP logic because of the addition of time. But, it still serves to increase confidence in the soundness of the logic by providing a kind of "sanity check."

Before considering the core meta-theorems, we must state a few lemmata that will be used in the following meta-theoretic proofs:

Lemma 3.1.

1. *If* $\Sigma; \Psi; \Gamma \Longrightarrow \gamma$, *then* $\Sigma, \Sigma'; \Psi, \Psi'; \Gamma, \Gamma' \Longrightarrow \gamma$.
2. *If* $\Sigma; \Psi, I \supseteq I''; \Gamma \Longrightarrow \gamma$ *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi, I' \supseteq I''; \Gamma \Longrightarrow \gamma$.
3. *If* $\Sigma; \Psi \models C$ *and* $\Sigma; \Psi, C; \Gamma \Longrightarrow \gamma$, *then* $\Sigma; \Psi; \Gamma \Longrightarrow \gamma$.
4. *If* $\Sigma \vdash t : s$ *and* $\Sigma, x{:}s; \Psi; \Gamma \Longrightarrow \gamma$, *then* $\Sigma; [t/x]\Psi; [t/x]\Gamma \Longrightarrow [t/x]\gamma$.

Proof. All parts follow by structural induction on the given derivation. □

As in GP logic, we are still interested in verifying the identity principle. However, with the shift in underlying judgments to hybrid, time-dependent forms, the statement of the theorem must change. For any proposition A, it should be possible to conclude from the hypothesis $A[I]$ that $A[I']$, provided I' is a subinterval of I. This generalizes the init rule, and is formalized in the following theorem.

Theorem 3.2 (Identity). *For all propositions* A, *if* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma, A[I] \Longrightarrow A[I']$.

Proof. By structural induction on A. □

A natural time-dependent property to expect of η_N logic is the notion of subsumption. For example, whenever one can prove that A is true on interval I, it should be possible, for any subinterval I' of I, to construct a similar proof that A is true on I'. This can be easily generalized to affirmations. Because this type of subsumption occurs in proof conclusions and not assumptions, it appears to the right of the $\Longrightarrow$ symbol in a hypothetical judgment. It is therefore termed right subsumption.

Theorem 3.3 (Right Subsumption).

1. *If* $\Sigma; \Psi; \Gamma \Longrightarrow A[I]$ *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma \Longrightarrow A[I']$.
2. *If* $\Sigma; \Psi; \Gamma \Longrightarrow (K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I$ *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma \Longrightarrow (K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I'$.

Proof. By simultaneous structural induction on the first given derivation. □

Subsumption can also occur for hypotheses. If interval I is a superinterval of I', the assumption that A is true on interval I is at least as powerful as assuming that A is true on I': the former assumption contains as much (and possibly more) information as the latter. Because hypotheses appear on the left side of the $\Longrightarrow$ symbol in a hypothetical judgment, this kind of subsumption is termed left subsumption.

Theorem 3.4 (Left Subsumption). *If* $\Sigma; \Psi; \Gamma, A[I'] \Longrightarrow \gamma$ *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma, A[I] \Longrightarrow \gamma$.

Proof. By nested induction on the structures of A and the first given derivation. □

Finally, we can reconsider cut elimination in the context of η_N logic. The admissibility of cut for the truth judgment remains relatively unchanged: a proof of $A[I]$ can replace the assumption $A[I]$ of any other proof. However, the admissibility of cut for the affirmation judgment changes in a significant way. As argued in the description of the $\langle\rangle L$ rule, an affirmation made by K during interval I is equivalent to truth, but only if we are currently reasoning about the beliefs that K holds during a subinterval I'. Thus, a proof of $(K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I$ can replace the assumption $A[I]$ in a proof of $(K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'$, provided that I is a superinterval of I'.

Theorem 3.5 (Admissibility of Cut).
1. *If* $\Sigma; \Psi; \Gamma \Longrightarrow A[I]$ *and* $\Sigma; \Psi; \Gamma, A[I] \Longrightarrow \gamma$, *then* $\Sigma; \Psi; \Gamma \Longrightarrow \gamma$.
2. *If* $\Sigma; \Psi; \Gamma \Longrightarrow (K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I$, $\Sigma; \Psi; \Gamma, A[I] \Longrightarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'$, *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma \Longrightarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'$.

Proof. By simultaneous nested induction on the structures of A and the given derivations. □

The above meta-theorems have been mechanically verified using the Twelf logical framework [35]. The Twelf proofs are available at `http://www.andrew.cmu.edu/user/hdeyoung/etalogic/twelf`.

3.3.2 Correspondence to GP Logic

Upon careful comparison of the rules of GP logic and the rules of η_N logic, a correspondence becomes evident. For example, consider the $\wedge R$ and $\langle\rangle L$ rules:

GP Logic	η_N **Logic**
$\dfrac{\Sigma; \Gamma \Longrightarrow A \quad \Sigma; \Gamma \Longrightarrow B}{\Sigma; \Gamma \Longrightarrow A \wedge B}\ \wedge R$	$\dfrac{\Sigma; \Psi; \Gamma \Longrightarrow A[I] \quad \Sigma; \Psi; \Gamma \Longrightarrow B[I]}{\Sigma; \Psi; \Gamma \Longrightarrow A \wedge B[I]}\ \wedge R$
$\dfrac{\Sigma; \Gamma, \langle K\rangle A, A \Longrightarrow K\ \mathsf{affirms}\ B}{\Sigma; \Gamma, \langle K\rangle A \Longrightarrow K\ \mathsf{affirms}\ B}\ \langle\rangle L$	$\dfrac{\Sigma; \Psi \models I \supseteq I' \quad \Sigma; \Psi; \Gamma, \langle K\rangle A[I], A[I] \Longrightarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'}{\Sigma; \Psi; \Gamma, \langle K\rangle A[I] \Longrightarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'}\ \langle\rangle L$

Because there are no notions of time or constraints in GP logic, the correspondence does not extend to these constructs.

The above intuition suggests that GP logic can be encoded into η_N logic. Let $\vec{I}$ denote a list of time intervals. Also, if $\Gamma = A_1, \dots, A_n$ is a GP logic context and $\vec{I} = I_1, \dots, I_n$, let $\Gamma[\vec{I}]$ be the η_N logic context $A_1[I_1], \dots, A_n[I_n]$.[2] Finally, define a translation for parameter contexts such that $\overline{\Sigma}$ is Σ with $\mathsf{interval}$ and time parameters removed.

This permits us to state the following theorem.

Theorem 3.6.
1. *If* $\overline{\Sigma}; \Gamma \Longrightarrow A$ *and* $\Sigma; \Psi \models I \supseteq I'$ *for all* $I \in \vec{I}$*, then* $\Sigma; \Psi; \Gamma[\vec{I}] \Longrightarrow A[I']$*.*
2. *If* $\overline{\Sigma}; \Gamma \Longrightarrow K\,\mathsf{affirms}\,A$ *and* $\Sigma; \Psi \models I \supseteq I'$ *for all* $I \in \vec{I}$*, then* $\Sigma; \Psi; \Gamma[\vec{I}] \Longrightarrow (K\,\mathsf{affirms}\,A)\,\mathsf{at}\,I'$*.*

Proof. By simultaneous structural induction on the first given derivation. □

Informally, this theorem states that by choosing appropriate intervals for the hypotheses and conclusion of a hypothetical judgment, it is possible to translate a valid GP logic derivation into a valid η_N logic derivation. In particular, if the lattice imposed by interval inclusion contains a greatest element, possibly written $(-\infty, \infty)$, then that greatest element can be used for each hypothesis and conclusion in the constructed η_N logic derivation.

It is natural to consider whether the converse of the above theorem holds. Specifically, is it possible to derive $\overline{\Sigma}; \Gamma \Longrightarrow A$ in GP logic and verify that $\Sigma; \Psi \models I \supseteq I'$ holds for each $I \in \vec{I}$, whenever $\Sigma; \Psi; \Gamma[\vec{I}] \Longrightarrow A[I']$ is derivable in η_N logic? (An analogous converse can be stated for affirmation consequents.)

The full converse does not hold: it is not the case that $\Sigma; \Psi \models I \supseteq I'$ for each $I \in \vec{I}$. When $\cdot; \cdot \not\models I_1 \supseteq I_2$, the following is a simple counterexample.

$$\frac{\cdot; \cdot \models I_2 \supseteq I_2}{\cdot; \cdot; A[I_1], P[I_2] \Longrightarrow P[I_2]}\ \mathsf{init}$$

However, as the following theorem shows, a partial converse does indeed hold.

Theorem 3.7.
1. *If* $\Sigma; \Psi; \Gamma[\vec{I}] \Longrightarrow A[I']$*, then* $\overline{\Sigma}; \Gamma \Longrightarrow A$*.*
2. *If* $\Sigma; \Psi; \Gamma[\vec{I}] \Longrightarrow (K\,\mathsf{affirms}\,A)\,\mathsf{at}\,I'$*, then* $\overline{\Sigma}; \Gamma \Longrightarrow K\,\mathsf{affirms}\,A$*.*

Proof. By simultaneous structural induction on the given derivation. □

Due to the difficulty of reasoning about context translations in Twelf and the existence of a translation from Γ to $\Gamma[\vec{I}]$ in these theorems, the above theorems have not been mechanically verified. This is an opportunity for future work.

[2]There is an implicit identity translation from GP logic propositions to η_N logic propositions here.

3.4 Conclusion

In this chapter, we have derived an authorization logic with explicit time from GP logic, as reviewed in Chapter 2. We illustrated its ability to model complex time-dependent policies through office entry and journal publication examples. We also studied the logic's meta-theory and established a correspondence to GP logic. In addition, we noted the logic's inability to model restrictions on the number of accesses. The effort to correct this deficiency is the focus of the next chapter.

Chapter 4

η_L Logic

In the previous chapter, we developed a non-linear authorization logic with explicit time and demonstrated its increased expressive power in the contexts of office entry and journal publication. In addition, we studied the logic's meta-theoretic properties and proved that it subsumes GP logic.

However, we also saw that even η_N logic is not sufficiently rich to model many natural access control policies. In particular, policies that place limits on the number of accesses or require finitely useable credentials could not be expressed in the logic.

To express such use-limited policies in a time-independent setting, previous work [23] has combined an authorization logic with linear logic, allowing the authorization constructs to model access control and the linear constructs to model usage limits. In this chapter, we follow this approach and combine η_N logic with linear logic to yield a logic that can model both time-dependent and use-limited policies.

Following a brief overview of linear logic, we present the formal system of η_L logic. The new logic's increased expressiveness is demonstrated through several examples. Finally, we study a few meta-theoretic properties of the logic, including the admissibility of cut.

4.1 An Overview of Linear Logic

Recall the office entry example from Section 3.2.1 in which an office owner could issue a credential to give a trusted colleague access to his office. For example, we described a scenario in which Alice could allow Bob to enter her office on only May 7, 2008 by issuing the credential:

$$\langle \text{Alice} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[5/7/08]$$

Bob could then use this credential in conjunction with the trust' policy to derive

$$\langle \text{admin} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[[t, t]]$$

for any time t during May 7, 2008. However, because the assumption corresponding to Alice's credential is persistent, Bob can enter her office an unlimited number of times on that day.

Instead, Alice would like to ensure that Bob can enter only once and only on that day. If the assumption corresponding to Alice's credential could somehow be consumed upon its first use, then it could not be used to authorize further accesses. A possible solution, then, is the extension of η_N logic with an ability to model the consumption of objects.

As a well-established logic for modeling such changes of state, linear logic [26, 17] is a perfect starting point. To model the consumption of objects, linear logic does not have a single notion of truth, but instead distinguishes truth into two classes: truths that must be used once, and only once, and truths that may be used zero or more times without restriction. A single-use truth corresponds to a resource, while a multi-use truth is akin to a fact: objects are ephemeral, but knowledge is persistent. It is also occasionally useful to think of an unrestricted truth as a resource factory that can produce an unlimited number of copies of a given resource.

Introducing such a refinement of truth affects the connectives found in linear logic. For example, implication splits into two forms. Linear implication, $\multimap$, can be applied to resources and facts. Unrestricted implication, $\supset$, on the other hand, can only be applied to facts. Conjunction splits into two forms as well. Simultaneous conjunction, $\otimes$, represents the existence of two resources in the same state: both resources can be had. On the other hand, alternative conjunction, &, represents a choice between two resources: both resources can be had, but only alternatively. As alternative conjunction represents an internal choice made by the reasoner, it is distinct from disjunction, $\oplus$, which corresponds to an *external* choice made by the environment.

4.2 Logical System

Now, we formally present η_L logic, making the above suggested combination of linear logic and η_N logic explicit. Changes are made to the system's judgments, and consequently its propositions and inference rules, while the first-order terms and sorts and constraints are unaffected.

4.2.1 First-Order Terms and Sorts

The system of first-order terms and sorts is carried over en bloc from η_N logic. Sorts $\mathsf{principal}$, time, and $\mathsf{interval}$ are still required, and the other sorts remain open-ended. Σ continues to stand for a context of parameters in scope ascribed with sorts. The judgment $\Sigma \vdash t{:}s$ still means that term t has sort s. Finally, we continue to write $[t/x]$ for the substitution of term t for all free occurrences of x.

4.2.2 Constraints

We also make no changes to the system of constraints. Superset constraints, $I \supseteq I'$, are still required, and other application-specific constraint forms may still be added as needed. Ψ continues to stand for a context of constraint assumptions. The constraint entailment judgment $\Sigma; \Psi \models C$ still states that constraint C holds, assuming the constraints in Ψ. Finally, although the constraint entailment judgment remains unspecified, the same six properties are required (repeated here for convenience).

(Hypothesis) $\Sigma; \Psi, C \models C$.
(Weakening) If $\Sigma; \Psi \models C$, then $\Sigma, \Sigma'; \Psi, \Psi' \models C$.
(Cut) If $\Sigma; \Psi \models C$ and $\Sigma; \Psi, C \models C'$, then $\Sigma; \Psi \models C'$.

(Substitution) If $\Sigma \vdash t{:}s$ and $\Sigma, x{:}s; \Psi \models C$, then $\Sigma; [t/x]\Psi \models [t/x]C$.
(Reflexivity) $\Sigma; \Psi \models I \supseteq I$.
(Transitivity) If $\Sigma; \Psi \models I \supseteq I'$ and $\Sigma; \Psi \models I' \supseteq I''$, then $\Sigma; \Psi \models I \supseteq I''$.

4.2.3 Judgments

In accordance with the goal of combining linear logic and η_N logic, we modify the judgments of η_N logic to become the resource-aware judgments of η_L logic. These modifications are analogous to the changes made to ordinary first-order logic to create linear logic.

In the transition from first-order logic to linear logic, truth forks into single-use truth and multi-use truth. In the same way, we split η_N logic's interval truth into single-use interval truth and multi-use interval truth. So, instead of having a judgment form meaning "Proposition A is true during interval I," we have two judgment forms: $A[I]$, meaning "Proposition A is a single-use truth (resource) during interval I," and $A[\![I]\!]$, meaning "Proposition A is a multi-use truth (fact) during interval I." Note that the syntax for the single-use interval truth judgment is the same as the syntax for the interval truth judgment of η_N logic. This should not cause confusion because the underlying logic should always be clear from the context.

To model principals' intents and policies, η_N logic contains an affirmation judgment form $(K \;\mathsf{affirms}\; A)\;\mathsf{at}\; I$, meaning "During interval I, principal K affirms the truth of proposition A on interval I." This judgment form is converted to the resource-aware $(K \;\mathsf{affirms}\; A)\;\mathsf{at}\; I$, meaning "During interval I, principal K affirms that proposition A is a single-use truth (resource) on interval I." This modification of the affirmation judgment for truth to an affirmation judgment for resources is based on the difference between linear GP logic [23] and (non-linear) GP logic [24]. Note that the single-use affirmation judgment uses the same syntax as the affirmation judgment of η_N logic; again, the underlying logic should be clear from the context.

As in η_N logic, η_L logic continues to use hypothetical judgments as the mechanism for handling assumptions. However, because the basic judgments have changed, it is necessary to reconsider the hypothetical judgment forms; in particular, it should be possible to assume both resources $A[I]$ and facts $A[\![I]\!]$.

The hypothetical judgment forms of linear η logic are:

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

where Σ is a context of parameters, ascribed with sorts, that may appear in Ψ, Γ, Δ, K, A, and I; Ψ is a context of constraints that are assumed to hold; Γ is a set of fact hypotheses of the form $A[\![I]\!]$; and Δ is a multiset of resource hypotheses of the form $A[I]$. In the following sections, we will write γ in place of the basic judgment to the right of $\Longrightarrow$ when its form does not matter.

The first of these hypothetical judgment forms means "Assuming that the constraints in Ψ hold, under the fact assumptions in Γ, and by using each resource assumption in Δ exactly once, resource A exists during interval I, parametrically in the members of Σ. Similarly, the second hypothetical judgment form means "Assuming that the constraints in Ψ hold, under the fact assumptions in Γ, and by using each resource assumption in Δ exactly once, principal K affirms, during interval I, that resource A exists on interval I, parametrically in the members of Σ."

4.2.4 Propositions

Now that the judgments of η_L logic have been introduced, it is possible to describe the propositions that these judgments act upon with the following grammar. We retain the propositions relating to affirmation, time, and constraints from η_N logic, but replace the propositions derived from ordinary first-order logic with those of linear logic.

$$\begin{aligned} A, B ::= {} & P \mid A \otimes B \mid \mathbf{1} \mid A \,\&\, B \mid \top \mid A \oplus B \mid A \multimap B \mid {!A} \mid A \supset B \mid \forall x{:}s.A \mid \exists x{:}s.A \\ & \mid \langle K \rangle A \mid A \mathbin{@} I \mid C \mathbin{\dot{\supset}} A \mid C \mathbin{\dot{\wedge}} A \end{aligned}$$

P stands for an arbitrary atomic proposition. $A \otimes B$, $A \,\&\, B$, $A \oplus B$, $A \multimap B$, and $A \supset B$ are simultaneous conjunction, alternative conjunction, disjunction, linear implication, and unrestricted implication, respectively, as described in Section 4.1. $\mathbf{1}$ is multiplicative truth, the unit for $\otimes$, and $\top$ is additive truth, the unit for &. $!A$ is the fact A encoded as a resource. $\forall x{:}s.A$ and $\exists x{:}s.A$ are universal and existential quantification, respectively. $\langle K \rangle A$ internalizes the affirmation judgment (K affirms A) at I. $A \mathbin{@} I$ internalizes the resource judgment $A[I]$. $C \mathbin{\dot{\supset}} A$ and $C \mathbin{\dot{\wedge}} A$ are constraint implication and constraint conjunction, respectively, adapted from Saranlı and Pfenning [38].

One might expect falsehood, $\mathbf{0}$, from linear logic to be included here. As for $\perp$ in η_N logic, any derivation of $\mathbf{0}$ would permit every principal to access every resource. Including $\mathbf{0}$ would therefore be a security risk. For this reason, we exclude $\mathbf{0}$ from η_L logic, as we did $\perp$ from η_N logic.

4.2.5 Inference Rules

Following the presentations of both GP logic and η_N logic, we now examine the inference rules for η_L logic. According to the philosophies of Gentzen [25] and Martin-Löf [32], these inference rules establish the logic's formal semantics.

We begin our discussion of the inference rules with the rule that defines the meaning of linear hypotheses:

$$\frac{\Sigma; \Psi \models I \supseteq I'}{\Sigma; \Psi; \Gamma; P[I] \Longrightarrow P[I']}\ \mathsf{init}$$

One would expect that, by assuming resource A exists during interval I, it should be possible to conclude that resource A exists during any subinterval I' of I. As in the previous logics, this property, restricted to atomic propositions P, is stated explicitly in the init rule. The rule is recovered in its full generality as Theorem 4.2 (cf. Section 4.4). Note that only the single resource hypothesis $P[I]$ is permitted in this rule. This ensures that resources cannot be discarded: since only $P[I]$ is used in this rule, any other resources that might have been allowed here would not have been used.

Next, we examine the inference rule that defines the meaning of fact hypotheses:

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

We previously mentioned that it is occasionally useful to think of facts as resource factories that can produce an unlimited number of resources of a given type. The copy rule is the most convincing example of this perspective. If the resource factory $A[\![I]\!]$ is assumed to exist, it can be called upon at any point to produce the resource $A[I]$. Producing this resource does not incapacitate the factory, and so the hypothesis $A[\![I]\!]$ persists in the copy rule's premise.

Next, we consider the right and left rules for the @ connective:

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

These rules have very similar structure to the corresponding rules of η_N logic (cf. Figure 3.1). There are only two differences. First, the rules now carry resource hypotheses, Δ, in addition to fact hypotheses, Γ. Second, the left rule operates on resource assumptions, not fact assumptions. These two differences are also present in the rules for other connectives borrowed from η_N logic.

Next, we consider the right and left rules for linear implication:

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

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

The judgment $A \multimap B[I]$ can be intuitively thought of as a plan for converting resource A to resource B that is only available during interval I. Such a conversion can be established by deriving $B[i]$ from $A[i]$, parametrically in the arbitrary subinterval i of I. The use of the fresh parameter i and constraint $I \supseteq i$ ensures that the conversion $A \multimap B$ is available at all times during I. This intuition is captured by the right rule, $\multimap R$. It is important to note that $A[i]$ is a *resource* hypothesis in the premise of this rule: $A[i]$ must be used exactly once in the derivation of $B[i]$.

The left rule, $\multimap L$, also supports the conversion intuition for $A \multimap B[I]$. Given the resource $A[I']$, the conversion can be carried out to produce the resource $B[I']$, provided I' is a subinterval of I. Observe that the resources are split among the premises of the rule; the resources Δ_1 used to establish $A[I']$ are consumed and cannot be used to establish γ in the other premise.

Next, we give the right and left rules for unrestricted implication:

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

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

These rules are quite similar to the $\multimap R$ and $\multimap L$ rules for linear implication. $A \supset B[I]$ may also be thought of as a conversion. However, the conversion is from *fact* A, not resource A, to resource B. For this reason, the $A[\![i]\!]$ assumption introduced in the $\supset R$ rule is a fact hypothesis. Accordingly, no resources may be used in establishing the requisite $A[I']$ in the $\supset L$ rule; the conversion only applies to facts. It should be noted that $A \supset B$ can be defined in terms of $\multimap$ and !, whose rules are given in Figure 4.1, as $(!A) \multimap B$; the $\supset R$ and $\supset L$ rules are derivable.

Finally, we present the right and left rules for simultaneous conjunction:

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

The judgment $A \otimes B[I]$ represents the existence of resources A and B in the same state during interval I. The right rule, $\otimes R$, shows that in establishing $A \otimes B[I]$, the resource hypotheses Δ_1, Δ_2 are split among the premises, with Δ_1 and Δ_2 being used to establish $A[I]$ and $B[I]$, respectively. By splitting the resources, the left rule, $\otimes L$, is justified: from $A \otimes B[I]$, we may have both $A[I]$ *and* $B[I]$ because the resources used to originally establish them were disjoint.

These rules and the rules for $\multimap$ imply the usual equivalence of curried and uncurried linear implications: $A \multimap (B \multimap C)[I]$ and $(A \otimes B) \multimap C[I]$ entail each other.

The remaining connectives are all standard to linear logic [26, 17]. Because they interact only weakly with time, their rules are rather straightforward modifications of the corresponding rules in linear logic. Figure 4.1 summarizes all of the inference rules of η_L logic.

Before concluding this section, we illustrate some key properties of η_L logic. We write $\Longrightarrow A$ if, for all Σ, Ψ, and I'', $\Sigma;\Psi;\cdot;\cdot \Longrightarrow A[I'']$ is derivable, and write $\not\Longrightarrow A$ otherwise. Also, $A \equiv B$ stands for $(A \multimap B) \,\&\, (B \multimap A)$.

First, we note that $\langle K\rangle$ remains similar to a lax modality [20] and that $\langle K\rangle A$ does not entail A in general:

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

Next, we give a few properties of the @ connective. These properties are analogous to those of η_N logic.

5. $\not\Longrightarrow (A \,@\, I) \multimap (A \,@\, I')$
6. $\Longrightarrow (I \supseteq I') \,\dot{\supset}\, ((A \,@\, I) \multimap (A \,@\, I'))$
7. $\Longrightarrow (A \,@\, I) \equiv (A \,@\, I \,@\, I')$
8. $\Longrightarrow ((A \otimes B) \,@\, I) \equiv ((A \,@\, I) \otimes (B \,@\, I))$
9. $\Longrightarrow ((A \,\&\, B) \,@\, I) \equiv ((A \,@\, I) \,\&\, (B \,@\, I))$
10. $\Longrightarrow ((A \oplus B) \,@\, I) \equiv ((A \,@\, I) \oplus (B \,@\, I))$
11. $\Longrightarrow ((A \multimap B) \,@\, I) \multimap ((A \,@\, I) \multimap (B \,@\, I))$
12. $\not\Longrightarrow ((A \,@\, I) \multimap (B \,@\, I)) \multimap ((A \multimap B) \,@\, I)$
13. $\Longrightarrow ((A \supset B) \,@\, I) \multimap ((A \,@\, I) \supset (B \,@\, I))$

$\boxed{\text{Basic Rules}}$

$$\frac{\Sigma; \Psi \models I \supseteq I'}{\Sigma; \Psi; \Gamma; P[I] \Longrightarrow P[I']}\ \textsf{init} \qquad \frac{\Sigma; \Psi; \Gamma, A\llbracket I \rrbracket; \Delta, A[I] \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma, A\llbracket I \rrbracket; \Delta \Longrightarrow \gamma}\ \textsf{copy}$$

$\boxed{A \mathbin{@} I}$

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

$\boxed{\text{Constraints}}$

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

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

$\boxed{\text{Affirmation and } \langle K \rangle A}$

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

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

$\boxed{\text{Other Connectives}}$

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

$$\frac{}{\Sigma; \Psi; \Gamma; \cdot \Longrightarrow \mathbf{1}[I]}\ \mathbf{1}R \qquad \frac{\Sigma; \Psi; \Gamma; \Delta \Longrightarrow \gamma}{\Sigma; \Psi; \Gamma; \Delta, \mathbf{1}[I] \Longrightarrow \gamma}\ \mathbf{1}L$$

Figure 4.1: The inference rules for η_L logic.

$\boxed{\text{Other Connectives, cont.}}$

$$\frac{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow A[I] \quad \Sigma;\Psi;\Gamma;\Delta \Longrightarrow B[I]}{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow A \,\&\, B[I]}\ \&R$$

$$\frac{\Sigma;\Psi;\Gamma;\Delta, A[I] \Longrightarrow \gamma}{\Sigma;\Psi;\Gamma;\Delta, A \,\&\, B[I] \Longrightarrow \gamma}\ \&L_1 \qquad \frac{\Sigma;\Psi;\Gamma;\Delta, B[I] \Longrightarrow \gamma}{\Sigma;\Psi;\Gamma;\Delta, A \,\&\, B[I] \Longrightarrow \gamma}\ \&L_2$$

$$\frac{}{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow \top[I]}\ \top R$$

$$\frac{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow A[I]}{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow A \oplus B[I]}\ \oplus R_1 \qquad \frac{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow B[I]}{\Sigma;\Psi;\Gamma;\Delta \Longrightarrow A \oplus B[I]}\ \oplus R_2$$

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

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

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

$$\frac{\Sigma;\Psi;\Gamma;\cdot \Longrightarrow A[I]}{\Sigma;\Psi;\Gamma;\cdot \Longrightarrow\, !A[I]}\ !R \qquad \frac{\Sigma;\Psi;\Gamma, A[\![I]\!];\Delta \Longrightarrow \gamma}{\Sigma;\Psi;\Gamma;\Delta, !A[I] \Longrightarrow \gamma}\ !L$$

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

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

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

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

Figure 4.2: The inference rules for η_L logic, continued.

14. $\not\Longrightarrow ((A \mathbin{@} I) \supset (B \mathbin{@} I)) \multimap ((A \supset B) \mathbin{@} I)$

15. $\not\Longrightarrow (((\langle K \rangle A) \mathbin{@} I) \multimap (\langle K \rangle (A \mathbin{@} I))$

16. $\not\Longrightarrow (\langle K \rangle (A \mathbin{@} I)) \multimap (((\langle K \rangle A) \mathbin{@} I)$

Property 5 shows that, in general, truth on one interval does not imply truth on another interval: the two intervals may be unrelated. However, as property 6 demonstrates, truth on an interval does indeed imply truth on a subinterval. Property 7 shows that only the innermost $@\, I$ qualification matters. Properties 8–10 indicate that the natural distributive laws for @ over $\otimes$, &, and $\oplus$ hold. Properties 11–14 show that @ distributes over $\multimap$ and $\supset$ only in one direction. Properties 15 and 16 demonstrate that @ does not distribute over $\langle K \rangle$ in either direction.

Finally, η_L logic is consistent:

17. $\not\Longrightarrow A$

4.3 Examples

To highlight the increased expressive power of η_L logic, we now present two examples that illustrate the use of finitely-usable credentials in combination with explicit time. First, we refine the office entry example from previous chapters for the new logic. Second, we consider the application of the logic to a simple homework assignment administration system.

In these examples, we adopt the conventions that $\otimes$, $\multimap$, $\supset$, and $\dot{\supset}$ are right associative, and that binding precendence decreases in the order: $\langle\rangle$; @; $\otimes$; $\multimap$, $\supset$, and $\dot{\supset}$; $\forall$.

4.3.1 Office Entry

Recall, from Section 3.2.1, the office entry example inspired by the Grey system [9, 8]. The example assumed an administrating principal, admin, that managed access to offices, and used the predicate $\mathsf{may\_enter}$, where $\mathsf{may\_enter}(K_2, K_1)$ meant that K_2 could enter K_1's office. The policies proposed for a PCA architecture based on η_N logic were:

$$\mathsf{own}' : \langle \text{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))[(-\infty, \infty)]$$

$$\begin{aligned}\mathsf{trust}' : \langle \text{admin} \rangle (\forall K_1{:}\mathsf{principal}.\forall K_2{:}\mathsf{principal}.\\ \langle K_1 \rangle \mathsf{may\_enter}(K_2, K_1) \supset\\ \mathsf{may\_enter}(K_2, K_1))[(-\infty, \infty)]\end{aligned}$$

But these policies did not model restrictions on the number of times that a user may enter an office. In fact, these restrictions cannot be expressed in η_N logic, motivating the design of η_L logic. Now that we have developed a logic ostensibly capable of modeling single-use authorizations, it should be possible to refine the own' and trust' policies to incorporate the desired usage restrictions. We begin by revising own'.

Because η_L logic contains $\langle K \rangle$ and $\forall$ propositions and uses the same interval structures as η_N logic, a first attempt at revision would be to use the same proposition and validity interval as own'. But, should the policy be represented as a fact or as a resource? Because it is natural to expect that an office owner should be able to enter his office an unlimited number of times (at least in the setting of academic offices), the policy should be represented as a fact:

$$\mathsf{own}'' : \langle \text{admin} \rangle (\forall K{:}\mathsf{principal}.\mathsf{may\_enter}(K, K))[\![(-\infty, \infty)]\!]$$

It is also possible to refine the trust' policy for use with η_L logic. Again, most of the policy can remain unchanged. However, because trust' uses implication, we must carefully choose the form of η_L logic implication. Since our goal is to permit office owners to restrict the number of times a colleague may enter, we choose $\multimap$: it can be applied to single-use resources, while $\supset$ can only be applied to multi-use facts. Since no usage restrictions should be placed on the trust'' policy itself, it is represented as the fact:

$$\begin{aligned}\mathsf{trust}'' : \langle \text{admin} \rangle (&\forall K_1{:}\mathsf{principal}.\forall K_2{:}\mathsf{principal}.\\ &\langle K_1 \rangle \mathsf{may\_enter}(K_2, K_1) \multimap\\ &\quad \mathsf{may\_enter}(K_2, K_1))[\![(-\infty, \infty)]\!]\end{aligned}$$

We now revisit the dilemma of Alice and Bob: Alice is away from the office on May 7, 2008 and wants to allow Bob to enter her office once (and only once) on that day. To do this, Alice can now issue the *single-use* credential, imported into the system as the resource hypothesis:

$$\mathcal{C}'' : \langle \text{Alice} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[5/7/08]$$

Suppose that Bob approaches Alice's office door at time t_0 and requests access. Provided that t_0 is during May 7, 2008, Bob can combine the trust'' policy with Alice's credential to construct the required proof of

$$\Sigma_{\text{A,B}}; \cdot; \mathsf{own}'', \mathsf{trust}''; \mathcal{C}'' \Longrightarrow \langle \text{admin} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[[t_0, t_0]]$$

where $\Sigma_{\text{A,B}}$ assigns the sort $\mathsf{principal}$ to all principals in the system. However, the $\multimap$ in the trust'' policy causes the resource hypothesis $\mathcal{C}''$ to be consumed. The reference monitor records this usage of $\mathcal{C}''$. If Bob attempts to enter Alice's office again at some later time t_1, he will be asked to prove

$$\Sigma_{\text{A,B}}; \cdot; \mathsf{own}'', \mathsf{trust}''; \cdot \Longrightarrow \langle \text{admin} \rangle \mathsf{may\_enter}(\text{Bob}, \text{Alice})[[t_1, t_1]]$$

The reference monitor does not allow the resource hypothesis $\mathcal{C}''$ to be used in this proof because it has already been consumed by the proof at time t_0. It is easy to check that it is impossible to prove the judgment required for access at time t_1. Thus, Alice has successfully restricted Bob to entering her office at most once during May 7, 2008.

As this example shows, η_L logic has improved η_N logic by permitting usage restrictions on authorizations. We proceed to give two further examples of the increased expressive power of η_L logic.

4.3.2 Filling Painkiller Prescriptions

We now consider the specification of pharmacy policies for dispensing painkilling medications in η_L logic. To prevent addiction, painkillers are tightly regulated. A patient must submit a valid doctor's prescription to the pharmacist and may only receive a few days worth of pills at a time.

Only after those pills are used can the patient be given more medication. The policies described below enforce these restrictions.

This example requires the application-specific sort int for integers, function symbols $+$ and $-$ for integer addition and subtraction, and a $\leq$ order constraint over integers. In addition, the following predicates are used:

$\mathsf{submit\_order}$	A request to submit a prescription.
$\mathsf{script}(K, n)$	A prescription for principal K to have n days worth of pills.
$\mathsf{is\_doctor}(D)$	Principal D is a doctor.
$\mathsf{record}(K, n)$	A pharmacist's record that principal K has n remaining days worth of pills on his prescription.
$\mathsf{receipt}(D, K, n, i)$	A receipt that a prescription signed by principal D for principal K to have n days worth of pills during interval i was received.
$\mathsf{request\_dispense}(n)$	A request that n days worth of pills be dispensed.
$\mathsf{pills}(K, n)$	Principal K has n days worth of pills.

Suppose that a doctor D wishes to issue a prescription for n days worth of painkilling medication to his patient K. He does so by issuing the consumable credential $\langle D\rangle(\mathsf{script}(K, n) \mathbin{@} i)[[t_i, \infty)]$. i is the interval over which the prescription is valid and medication may be dispensed. t_i is the time at which the doctor signs the prescription. Note that the left endpoint of i need not match t_i; for example, the doctor may sign the prescription several days before the surgery for which the medication is needed.

The first policy, order, specifies the procedure for submitting a doctor's prescription to a pharmacy:

$$\begin{aligned}\mathsf{order} : (&\forall K{:}\mathsf{principal}.\forall t{:}\mathsf{time}.\forall D{:}\mathsf{principal}.\forall n{:}\mathsf{int}.\forall i{:}\mathsf{interval}.\\ &\langle K\rangle \mathsf{submit\_order} \mathbin{@} [t, t] \multimap\\ &\langle D\rangle(\mathsf{script}(K, n) \mathbin{@} i) \mathbin{@} [t, t] \multimap\\ &\langle P\rangle \mathsf{is\_doctor}(D) \mathbin{@} [t, t] \supset\\ &\quad(\langle P\rangle(\mathsf{record}(K, n) \mathbin{@} i) \mathbin{@} [t, \infty) \otimes\\ &\quad\ \langle P\rangle \mathsf{receipt}(D, K, n, i) \mathbin{@} [t, \infty)))[\![(-\infty, \infty)]\!]\end{aligned}$$

Principal K begins a transaction at time t by creating the single-use credential $\langle K\rangle\mathsf{submit\_order}[[t, t]]$. The pharmacy P accepts the prescription and issues a receipt if the following conditions are met:

1. $\langle D\rangle(\mathsf{script}(K, n) \mathbin{@} i) \mathbin{@} [t, t]$—at time t, there must exist a prescription for K to have some medication over some interval.
2. $\langle P\rangle\mathsf{is\_doctor}(D) \mathbin{@} [t, t]$—the pharmacy P must verify that principal D is indeed a certified doctor at time t.

Under these conditions, the pharmacy will create an internal record of the prescription so that medication can be dispensed by issuing the credential $\langle P\rangle(\mathsf{record}(K, n) \mathbin{@} i) \mathbin{@} [t, \infty)$. In addition, the pharmacy will give K a receipt of the transaction, modeled as $\langle P\rangle\mathsf{receipt}(D, K, n, i) \mathbin{@} [t_i, \infty)$.

The second policy of the system, $\mathsf{dispense}$, specifies the conditions under which medication can be dispensed:

$$\begin{aligned}
\mathsf{dispense} : (&\forall K{:}\mathsf{principal}.\forall n'{:}\mathsf{int}.\forall t{:}\mathsf{time}.\forall P{:}\mathsf{principal}.\forall n{:}\mathsf{int}.\forall t_f{:}\mathsf{time}.\\
&\quad \langle K\rangle \mathsf{request\_dispense}(n') \mathbin{@} [t,t] \multimap\\
&\quad \langle P\rangle(\mathsf{record}(K,n) \mathbin{@} [t,t_f]) \mathbin{@} [t,t] \multimap\\
&\quad (n' > 0) \mathbin{\dot{\supset}} (n' \le n) \mathbin{\dot{\supset}} (n' \le 7) \mathbin{\dot{\supset}}\\
&\qquad (\mathsf{pills}(K,n') \mathbin{@} [t,\infty) \otimes\\
&\qquad\ \langle P\rangle(\mathsf{record}(K,n-n') \mathbin{@} [t+n',t_f]) \mathbin{@} [t,\infty)))[\![(-\infty,\infty)]\!]
\end{aligned}$$

A principal K requests that n' days worth of medication be dispensed at time t by creating the single-use credential $\langle K\rangle \mathsf{request\_dispense}(n')[[t,t]]$. The pharmacy P carries out this request and updates its internal record if the following conditions are met:

1. $\langle P\rangle(\mathsf{record}(K,n) \mathbin{@} [t,t_f]) \mathbin{@} [t,t]$—at time t, the pharmacy has a record that K may be given n days worth of medication during $[t,t_f]$.
2. $n' > 0$—the number of days worth of medication requested is positive. This prevents negative requests that would increase the amount listed in the pharmacy's record.
3. $n' \le n$—the number of days worth of medication requested is no more than the total remaining amount K may have. This prevents K from exceeding his prescribed amount.
4. $n' \le 7$—the number of days worth of medication requested is no more than 7. This ensures that at most one week's worth of medication is dispensed at one time.

Provided that these conditions are satisfied, the pharmacy will dispense n' pills to K. K possesses these pills from time t onward. The pharmacy also updates its record by deducting n' from the number of pills remaining on K's prescription. In addition, the interval over which K may request these remaining pills is changed to $[t+n',t_f]$. The expiration date remains the same, but the left endpoint is moved so that K must wait n' more days before new pills can be dispensed. This controls the average rate at which K can consume the pills.

Observe that it is critical that the pharmacy's record is a consumable credential. If the record was persistent, it would be impossible to accurately deduct medication dispensed and adjust the validity interval: one could use the old record to obtain more medication than prescribed.

4.3.3 A Homework Assignment Administration System

In this example, we consider the application of η_L logic to a homework assignment administration system. Time is used to express the release and due dates of assignments, while linearity is used to model changes of state in the system.

We postulate sorts for assignments and courses: assignment and course, respectively. In addition, we introduce the following predicates:

$\mathsf{request\_view}(A,C)$	A request to view assignment A of course C.
$\mathsf{request\_submit}(A,C)$	A request to submit answers for assignment A of course C.
$\mathsf{is\_professor}(P,C)$	P is a professor for course C.
$\mathsf{is\_student}(S,C)$	S is a student enrolled in course C.
$\mathsf{is\_assignment}(A,C)$	A is an asignment for the students in course C.
$\mathsf{may\_view}(S,A,C)$	S may view assignment A of course C.
$\mathsf{may\_submit}(S,A,C)$	S may submit answers for assignment A of course C.
$\mathsf{change\_date}(A,C,t'_r,t'_d)$	A request to change the release and due dates for assignment A of course C to t'_r and t'_d, respectively.

We also assume an administrating principal, admin, that manages the system. This administrator is responsible for initializing the courses in the system at the beginning of each semester. First, the administrator issues multi-use certificates identifying the instructors of each course. For example, if principal P is the professor for course C during Spring 2008, the administrator issues $\langle \text{admin} \rangle \mathsf{is\_professor}(P, C)[\![\text{S}'08]\!]$. Second, the administrator issues a multi-use certificate for each student enrolled in each course. For example, if principal S is a student enrolled in course C during Spring 2008, the administrator issues $\langle \text{admin} \rangle \mathsf{is\_student}(S, C)[\![\text{S}'08]\!]$.

During the semester, a professor P can create an assignment A for course C with release date t_r and due date t_d by stating $\langle P \rangle \mathsf{is\_assignment}(A, C)[[t_r, t_d]]$. Note that this is a single-use credential. This is done to permit the professor to change the release and due dates, if desired (using the change policy).

The system's first policy specifies the conditions under which a principal may view an assignment:

$$\begin{array}{l}
\mathsf{view} : (\forall S{:}\mathsf{principal}.\forall A{:}\mathsf{assignment}.\forall C{:}\mathsf{course}. \\
\quad \forall t{:}\mathsf{time}.\forall P{:}\mathsf{principal}.\forall t_r{:}\mathsf{time}.\forall t_d{:}\mathsf{time}. \\
\quad\quad \langle S \rangle \mathsf{request\_view}(A, C) \mathbin{@} [t, t] \multimap \\
\quad\quad \langle \text{admin} \rangle \mathsf{is\_student}(S, C) \mathbin{@} [t, t] \supset \\
\quad\quad \langle P \rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d] \multimap \\
\quad\quad \langle \text{admin} \rangle \mathsf{is\_professor}(P, C) \mathbin{@} [t_r, t_d] \supset \\
\quad\quad (t \geq t_r) \mathbin{\dot{\supset}} \\
\quad\quad\quad (\langle \text{admin} \rangle \mathsf{may\_view}(S, A, C) \mathbin{@} [t, t] \otimes \\
\quad\quad\quad\quad \langle P \rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d]))[\![(-\infty, \infty)]\!]
\end{array}$$

A principal S can make a request to view assignment A for course C at time t by creating the certificate $\langle S \rangle \mathsf{request\_view}(A, C)[[t, t]]$. The adminstrator will let S view the assignment at time t, represented as $\langle \text{admin} \rangle \mathsf{may\_view}(S, A, C) \mathbin{@} [t, t]$, if the following four conditions are met:

1. $\langle \text{admin} \rangle \mathsf{is\_student}(S, C) @ [t, t]$—the administrator affirms that principal S is a student enrolled in course C at the time the request is made, that is, at time t.
2. $\langle P \rangle \mathsf{is\_assignment}(A, C) @ [t_r, t_d]$—some principal P affirms that A is an assignment for course C with release and due dates t_r and t_d, respectively.
3. $\langle \text{admin} \rangle \mathsf{is\_professor}(P, C) @ [t_r, t_d]$—the administrator affirms that the above principal P is actually an instructor for course C for the duration the assignment, that is, during $[t_r, t_d]$.
4. $t \geq t_r$—the time at which student S requests access is after the assignment has been released. This prevents students from viewing a draft assignment.

Note that the $\mathsf{is\_assignment}$ credential is consumed and immediately regenerated by this policy. It must be a consumable credential to facilitate the changing of the assignment release and due dates, and yet it cannot be *permanently* consumed in this policy because then the first viewing would destroy the assignment.

The system also includes a policy that describes how a principal may submit answers to an assignment:

$$\begin{aligned}
\mathsf{submit} : (&\forall S{:}\mathsf{principal}.\forall A{:}\mathsf{assignment}.\forall C{:}\mathsf{course}.\\
&\forall t{:}\mathsf{time}.\forall P{:}\mathsf{principal}.\forall t_r{:}\mathsf{time}.\forall t_d{:}\mathsf{time}.\\
&\langle S\rangle \mathsf{request\_submit}(A, C) \mathbin{@} [t, t] \multimap\\
&\langle \mathrm{admin}\rangle \mathsf{is\_student}(S, C) \mathbin{@} [t, t] \supset\\
&\langle P\rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d] \multimap\\
&\langle \mathrm{admin}\rangle \mathsf{is\_professor}(P, C) \mathbin{@} [t_r, t_d] \supset\\
&(t \in [t_r, t_d]) \dot{\supset}\\
&\quad (\langle \mathrm{admin}\rangle \mathsf{may\_submit}(S, A, C) \mathbin{@} [t, t] \otimes\\
&\quad\ \langle P\rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d]))[\![(-\infty, \infty)]\!]
\end{aligned}$$

This policy is quite similar to the view policy. A principal S signals his intent to submit answers for assignment A in course C at time t by constructing the credential $\langle S\rangle\mathsf{request\_submit}(A, C)[[t, t]]$. The administrator allows S to submit answers at time t, represented in the submit policy as $\langle \mathrm{admin}\rangle\mathsf{may\_submit}(S, A, C) \mathbin{@} [t, t]$, if four conditions are met. The first three conditions are the same as those for the view policy. The fourth condition is $t \in [t_r, t_d]$, that is, the time of request must be before the assignment due date (and after the release date). This prevents late assignments from being submitted. Note that, as in the view policy, the is_assignment credential is regenerated.

The final policy permits a course professor to change assignment release and due dates:

$$\begin{aligned}
\mathsf{change} : (&\forall P{:}\mathsf{principal}.\forall A{:}\mathsf{assignment}.\forall C{:}\mathsf{course}.\\
&\forall t'_r{:}\mathsf{time}.\forall t'_d{:}\mathsf{time}.\forall t_r{:}\mathsf{time}.\forall t_d{:}\mathsf{time}.\\
&\langle P\rangle \mathsf{change\_date}(A, C, t'_r, t'_d) \multimap\\
&\langle P\rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d] \multimap\\
&\langle \mathrm{admin}\rangle \mathsf{is\_professor}(P, C) \supset\\
&\quad \langle P\rangle \mathsf{is\_assignment}(A, C) \mathbin{@} [t'_r, t'_d])[\![(-\infty, \infty)]\!]
\end{aligned}$$

A principal P can initiate the process of changing the release and due dates of assignment A for course C to t'_r and t'_d, respectively, by issuing the credential $\langle P\rangle\mathsf{change\_date}(A, C, t'_r, t'_d)$. If the same principal P has already declared A to be an assignment for course C, represented in the change policy as $\langle P\rangle\mathsf{is\_assignment}(A, C) \mathbin{@} [t_r, t_d]$, and if the administrator affirms P to be a professor for course C, represented as $\langle \mathrm{admin}\rangle\mathsf{is\_professor}(P, C)$, then the dates will be changed. This policy finally justifies the decision to make is_assignment credentials usable only once: if the credential was persistent, then the assignment would have two release dates and two due dates.

4.4 Meta-theory

Now that η_L logic has been formally described and its increased expressive power has been illustrated by examples, we turn to a study of its meta-theoretic properties. As for previous logics, we show that natural properties of η_L logic indeed hold, to increase confidence in the logic's foundations and demonstrate its soundness.

Before considering any meta-theorems that are interesting in their own right, we must state a few lemmata:

Lemma 4.1.

1. If $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow \gamma$*, then* $\Sigma, \Sigma'; \Psi, \Psi'; \Gamma, \Gamma'; \Delta \Longrightarrow \gamma$.

2. *If* $\Sigma; \Psi, I \supseteq I''; \Gamma; \Delta \Longrightarrow \gamma$ *and* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi, I' \supseteq I''; \Gamma; \Delta \Longrightarrow \gamma$.
3. *If* $\Sigma; \Psi \models C$ *and* $\Sigma; \Psi, C; \Gamma; \Delta \Longrightarrow \gamma$*, then* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow \gamma$.
4. *If* $\Sigma \vdash t : s$ *and* $\Sigma, x{:}s; \Psi; \Gamma; \Delta \Longrightarrow \gamma$*, then* $\Sigma; [t/x]\Psi; [t/x]\Gamma; [t/x]\Delta \Longrightarrow [t/x]\gamma$.

Proof. All parts follow by structural induction on the given derivation. □

In the presentation of η_N logic's meta-theory (cf. Section 3.3.1), we studied an identity principle that generalized the init rule from atomic propositions to compound propositions. It is possible to make the same generalization in η_L logic. As the following theorem shows, from the assumption that proposition A is a resource during interval I, it is possible to conclude that A is a resource during any subinterval I' of I. Because the theorem concerns resource hypotheses and each resource hypothesis must be used exactly once, no other resource hypotheses are permitted here.

Theorem 4.2 (Identity)**.** *For all propositions* A*, if* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi; \Gamma; A[I] \Longrightarrow A[I']$.

Proof. By structural induction on A. □

In the presentation of η_N logic, we also examined subsumption as a meta-theoretic property. The most basic form of subsumption occurred on the right side of $\Longrightarrow$: from a proof that A is true on interval I, we were able to construct a similar proof that A is true on a subinterval I'. This notion of right subsumption from η_N logic can be easily extended to η_L logic; the theorem and proof are no more complicated.

Theorem 4.3 (Right Subsumption)**.**
1. *If* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I]$ *and* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow A[I']$.
2. *If* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I$ *and* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K \text{ affirms } A) \text{ at } I'$.

Proof. By simultaneous structural induction on the first given derivation. □

For η_N logic, we also considered subsumption on the left of $\Longrightarrow$. We were able to replace an assumption that A is true on interval I' with an assumption that A is true on a superinterval I. It is also possible to extend this notion of left subsumption to η_L logic. Because the logic now includes fact hypotheses, the theorem must be expanded to make an analogous statement about fact hypotheses.

Theorem 4.4 (Left Subsumption)**.**
1. *If* $\Sigma; \Psi; \Gamma; \Delta, A[I'] \Longrightarrow \gamma$ *and* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi; \Gamma; \Delta, A[I] \Longrightarrow \gamma$.
2. *If* $\Sigma; \Psi; \Gamma, A[\![I']\!]; \Delta \Longrightarrow \gamma$ *and* $\Sigma; \Psi \models I \supseteq I'$*, then* $\Sigma; \Psi; \Gamma, A[\![I]\!]; \Delta \Longrightarrow \gamma$.

Proof. By simultaneous nested induction on the structures of A and the first given derivation. □

Finally, we examined the admissibility of cut in our study of the meta-theory of η_N logic. We can also establish the admissibility of cut for η_L logic. However, two significant modifications must be made to the theorem statement. First, when replacing the resource hypothesis $A[I]$ with a proof of $A[I]$, we must be careful to respect the single-use nature of resources. For this reason, the cut rules join the distinct multi-sets of resources, Δ and Δ', used by the two proofs being combined. Second, a cut rule for fact hypotheses is needed. The proof used to replace a fact hypothesis must prove a fact, and therefore cannot contain resource hypotheses.

Theorem 4.5 (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* $\Sigma; \Psi; \Gamma; \Delta \Longrightarrow (K\,\mathsf{affirms}\,A)\,\mathsf{at}\,I$, $\Sigma; \Psi; \Gamma; \Delta', A[I] \Longrightarrow (K\,\mathsf{affirms}\,B)\,\mathsf{at}\,I'$, *and* $\Sigma; \Psi \models I \supseteq I'$, *then* $\Sigma; \Psi; \Gamma; \Delta', \Delta \Longrightarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'$.

Proof. By simultaneous nested induction on the structures of A and the given derivations. □

Because of the difficulty of encoding the meta-theory of linear logics in Twelf, we have not attempted to mechanically verify the above theorems, instead relying on traditional pencil-and-paper proofs. However, Reed's work on hybrid LF [37] shows promising preliminary steps toward a framework for mechanically verifying linear meta-theorems. We expect that verifying the meta-theory of η_L logic would be a straightforward exercise in such a framework.

In Section 3.3.2, we established a formal correspondence between η_N logic and GP logic, in addition to the metatheory for η_N logic itself. It is also possible to establish a correspondence between η_L logic and a linear version of GP logic (as in [23]). However, in the interest of space, we will not present a linear GP logic, and therefore do not state the correspondence theorem.

4.5 Conclusion

In this chapter, we revised η_N logic to account for access control policies that require finitely usable credentials, creating η_L logic. We illustrated the new logic's increased expressiveness through two examples: office entry and homework administration. Finally, we conducted a small meta-theoretic study of the logic, culminating in a proof of the admissibility of cut. In the following chapter, we continue to discuss η_L logic by constructing an alternative, natural deduction formulation of the logic and a simple proof checker for it.

Chapter 5

A Proof Checker for η_L Logic

In the previous chapter, we developed a linear authorization logic with explicit time by naturally modifying η_N logic to possess judgments for single-use resources and multi-use facts. We saw that η_L logic is sufficiently expressive to model authorization policies that require single-use credentials or other mutable state.

Proof-carrying authorization (PCA) [6, 7] is an appealing mechanism for enforcing such policies in practice. In PCA, the reference monitor for a given resource requests a proof of authorization for each access request. Only after the reference monitor has verified the proof's correctness will access be granted. For this reason, a sound (and complete) proof checker is a critical component of any PCA architecture.

As the first small step toward a PCA architecture based on η_L logic, this chapter concerns the implementation of a proof checker for the logic. First, we present the well-formedness rules for terms, constraints, and propositions that have been deferred by earlier chapters. Next, we introduce proof terms and describe a bidirectional type checking system built from the logic. Finally, we discuss the implementation techniques used in handling the diverse aspects of the logic, most notably linearity and constraints.

A proof-of-concept implementation of the proof checker presented in this chapter is available online at `http://www.andrew.cmu.edu/user/hdeyoung/etalogic/checker`.

5.1 Formal Proof Checker

5.1.1 Sorts, Function Symbols, and Predicates

We retain from η_L logic the distinguished sorts principal, time, and interval for principals, times, and intervals (sets of time), respectively. The language of sorts can still be extended with application-specific sorts, but this open-endedness must now be made explicit for the purpose of implementation. Sort constants α are introduced to this end, completing the language of sorts:

$$s ::= \mathsf{principal} \mid \mathsf{time} \mid \mathsf{interval} \mid \alpha$$

To list the available sort constants, a signature is needed:

$$\Phi ::= \cdot \mid \Phi, \alpha{:}\mathbf{sort} \mid \ \ldots$$

$\boxed{\vdash \Phi\ \mathbf{ok}}$

$$\frac{}{\vdash \cdot\ \mathbf{ok}} \qquad \frac{\vdash \Phi\ \mathbf{ok} \quad \alpha{:}\mathbf{sort} \notin \mathrm{Dom}(\Phi)}{\vdash \Phi, \alpha{:}\mathbf{sort}\ \mathbf{ok}}$$

$$\frac{\Phi \vdash s_i : \mathbf{sort} \text{ for all } 1 \leq i \leq n \quad \Phi \vdash s' : \mathbf{sort} \quad f \notin \mathrm{Dom}(\Phi)}{\vdash \Phi, f{:}(s_1 \times \cdots \times s_n) \rightarrow s'\ \mathbf{ok}}$$

$$\frac{\Phi \vdash s_i : \mathbf{sort} \text{ for all } 1 \leq i \leq n \quad p \notin \mathrm{Dom}(\Phi)}{\vdash \Phi, p{:}(s_1 \times \cdots \times s_n)\ \mathbf{ok}}$$

$\boxed{\Phi \vdash s : \mathbf{sort}}$

$$\frac{\vdash \Phi\ \mathbf{ok}}{\Phi \vdash \mathsf{principal} : \mathbf{sort}} \qquad \frac{\vdash \Phi\ \mathbf{ok}}{\Phi \vdash \mathsf{time} : \mathbf{sort}} \qquad \frac{\vdash \Phi\ \mathbf{ok}}{\Phi \vdash \mathsf{interval} : \mathbf{sort}} \qquad \frac{\vdash \Phi\ \mathbf{ok} \quad \Phi(\alpha) = \mathbf{sort}}{\Phi \vdash \alpha : \mathbf{sort}}$$

Figure 5.1: The well-formedness rules for signatures and sorts.

In fact, sort constants and the signatures declaring them were implicitly assumed in the presentation of η_L logic in Chapter 4. However, because the available sort constants remain unchanged throughout a derivation, we avoided introducing them explicitly to simplify the initial discussion.

Like sort constants, function symbols and predicates were glossed over in Chapter 4 to avoid obscuring the core of η_L logic. Because the available function symbols and predicates also remain unchanged throughout a derivation, they, too, can be included in the signature Φ. We write $f{:}(s_1 \times \cdots \times s_n) \rightarrow s'$ to indicate that f is a function symbol taking n arguments of sorts $s_1, \ldots, s_n$ and returning a term of sort s', and write $p{:}(s_1 \times \cdots \times s_n)$ to show that p is a predicate on n terms of sorts $s_1, \ldots, s_n$.

$$\Phi ::= \cdot \mid \Phi, \alpha{:}\mathbf{sort} \mid \Phi, f{:}(s_1 \times \cdots \times s_n) \rightarrow s' \mid \Phi, p{:}s_1 \times \cdots \times s_n$$

The judgment $\vdash \Phi\ \mathbf{ok}$ means that Φ is a well-formed signature; its rules are given in Figure 5.1.

We write $\Phi \vdash s : \mathbf{sort}$ for the judgment that s is a well-formed sort in the signature Φ. Figure 5.1 gives the rules for this judgment. The predefined sorts are well-formed in any context and a sort constant is well-formed in Φ if it appears in Φ.

5.1.2 Terms

Although the representation of principals, times, and intervals is left unspecified in η_L logic, our proof checker must fix constructors for these terms. We choose to give no explicit constructors for principals, and instead rely on term parameters as the sole source of principals. In keeping with the examples from Chapters 3 and 4, we let time constants range over integers ($\overline{n}$) and positive and negative infinities (∞ and $-\infty$). Similarly, an interval is constructed as a pair of terms ($[t, t']$). We also allow the application of function symbols to a list of terms. This leads to the following

$\boxed{\Phi \vdash \Sigma \ \mathbf{ok}}$

$$\frac{\vdash \Phi \ \mathbf{ok}}{\Phi \vdash \cdot \ \mathbf{ok}} \qquad \frac{\Phi \vdash \Sigma \ \mathbf{ok} \quad x \notin \mathrm{Dom}(\Sigma) \quad \Phi \vdash s : \mathbf{sort}}{\Phi \vdash \Sigma, x{:}s \ \mathbf{ok}}$$

$\boxed{\Phi; \Sigma; \Psi \vdash t : s}$

$$\frac{\Phi; \Sigma \vdash \Psi \ \mathbf{ok}}{\Phi; \Sigma; \Psi \vdash \overline{n} : \mathsf{time}} \qquad \frac{\Phi; \Sigma \vdash \Psi \ \mathbf{ok}}{\Phi; \Sigma; \Psi \vdash \infty : \mathsf{time}} \qquad \frac{\Phi; \Sigma \vdash \Psi \ \mathbf{ok}}{\Phi; \Sigma; \Psi \vdash -\infty : \mathsf{time}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash t : \mathsf{time} \quad \Phi; \Sigma; \Psi \vdash t' : \mathsf{time} \quad \Phi; \Sigma; \Psi \models t' \geq t}{\Phi; \Sigma; \Psi \vdash [t, t'] : \mathsf{interval}}$$

$$\frac{\Phi(f) = (s_1 \times \cdots \times s_n) \to s' \quad \Phi; \Sigma; \Psi \vdash t_i : s_i \text{ for all } 1 \leq i \leq n}{\Phi; \Sigma; \Psi \vdash f(t_1, \ldots, t_n) : s'} \qquad \frac{\Phi; \Sigma \vdash \Psi \ \mathbf{ok} \quad \Sigma(x) = s}{\Phi; \Sigma; \Psi \vdash x : s}$$

Figure 5.2: The well-formedness rules for parameter contexts and terms.

language of terms:

$$t ::= \overline{n} \mid \infty \mid -\infty \mid [t, t'] \mid f(t_1, \ldots, t_n) \mid x$$

As in earlier chapters, we require an unordered context of term parameters ascribed with sorts to track the parameters in scope:

$$\Sigma ::= \cdot \mid \Sigma, x{:}s$$

The well-formedness rules for contexts of parameters are given in Figure 5.2.

With the language of terms defined, we should now describe the conditions under which terms are well-sorted. It is natural to assume that the time constructors $\overline{n}$, ∞, and $-\infty$ have sort time under any conditions, as they do not include parameters or any complex structure. Also, we assign the codomain sort of function symbol f to $f(t_1, \ldots, t_n)$ if $t_1, \ldots,$ and t_n have the domain sorts of f. Finally, it is standard practice to assign sort s to parameter x if this assignment is given in the parameter context Σ.

However, the conditions under which the term $[t, t']$ should have sort $\mathsf{interval}$ are not as clear. We intend that both t and t' have sort time. If this is adopted as the *only* condition for well-sortedness, how should $[t, t']$ be interpreted? There are at least two options.

We could interpret $[t, t']$ as an *unordered* pair of times that corresponds to the set of times between $\min\{t, t'\}$ and $\max\{t, t'\}$. Or, we could interpret $[t, t']$ as the empty set whenever t is larger than t'.

Neither option is particularly compelling, as they do not correspond to typical interpretations of mathematical intervals. It seems cleaner to instead require that the left endpoint of an interval be no larger than the right endpoint. Ordering constraints $t' \geq t$ from a context Ψ (cf. Section 5.1.3) are therefore needed during sorting. We write $\Phi; \Sigma; \Psi \vdash t : s$ for the judgment that t has sort s. Figure 5.2 gives its rules.

$\boxed{\Phi; \Sigma; \Psi \vdash C \textbf{ constraint}}$

$$\frac{\Phi; \Sigma; \Psi \vdash t : \mathsf{time} \quad \Phi; \Sigma; \Psi \vdash t' : \mathsf{time}}{\Phi; \Sigma; \Psi \vdash t \geq t' \textbf{ constraint}}$$

$\boxed{\Phi; \Sigma \vdash \Psi \textbf{ ok}}$

$$\frac{\Phi \vdash \Sigma \textbf{ ok}}{\Phi; \Sigma \vdash \cdot \textbf{ ok}} \qquad \frac{\Phi; \Sigma; \Psi \vdash C \textbf{ constraint}}{\Phi; \Sigma \vdash \Psi, C \textbf{ ok}}$$

Figure 5.3: The well-formedness rule for constraints and constraint contexts.

5.1.3 Constraints

Only ordering constraints $t \geq t'$ on times are required for the proof checker implementation. Although it would be relatively straightforward to introduce other constraints, we do not choose to do so. The language of constraints is therefore simply:

$$C ::= t \geq t'$$

Superset constraints $I \supseteq I'$ are not needed for the proof checker: as discussed in Section 5.2.4, they can be translated to a conjunction of $\geq$ constraints. Keeping this in mind, we continue to use $I \supseteq I'$ to simplify the notation.

The judgment $\Phi; \Sigma; \Psi \vdash C$ **constraint** means that constraint C is well-formed. The ordering constraint $t \geq t'$ is well-formed if both t and t' are well-formed times. This rule is given in Figure 5.3.

As alluded to in previous rules, a context of constraints is needed:

$$\Psi ::= \cdot \mid \Psi, C$$

We write $\Phi; \Sigma \vdash \Psi$ **ok** when Ψ is a well-formed constraint context. Figure 5.3 gives the rules for this judgment.

We continue to write $\Phi; \Sigma; \Psi \models C$ for the judgment that constraint C holds. In the following presentation of bidirectional type checking, the constraint solver is taken as a black box. The specific decision procedure that was implemented is discussed in Section 5.2.4.

5.1.4 Propositions and Types

The propositional connectives are retained from η_L logic, but we refine the description of atomic propositions. An atomic proposition P is now a predicate p applied to a list of terms $t_1, \ldots, t_n$: $p(t_1, \ldots, t_n)$. The language of propositions is summarized by the following grammar:

$$\begin{aligned} A, B ::= {} & p(t_1, \ldots, t_n) \mid A \otimes B \mid \mathbf{1} \mid A \,\&\, B \mid \top \mid A \oplus B \mid A \multimap B \mid {!A} \mid A \supset B \mid \forall x{:}s.A \mid \exists x{:}s.A \\ & \mid \langle K \rangle A \mid A \mathbin{@} I \mid C \dot{\supset} A \mid C \dot{\wedge} A \end{aligned}$$

$\boxed{\Phi; \Sigma; \Psi \vdash A \textbf{ prop}}$

$$\frac{\Phi(p) = s_1 \times \cdots \times s_n \quad \Phi; \Sigma; \Psi \vdash t_i : s_i \text{ for all } 1 \leq i \leq n}{\Phi; \Sigma; \Psi \vdash p(t_1, \ldots, t_n) \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash B \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash A \otimes B \textbf{ prop}}$$

$$\frac{\Phi; \Sigma \vdash \Psi \textbf{ ok}}{\Phi; \Sigma; \Psi \vdash \mathbf{1} \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash B \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash A \,\&\, B \textbf{ prop}} \qquad \frac{\Phi; \Sigma \vdash \Psi \textbf{ ok}}{\Phi; \Sigma; \Psi \vdash \top \textbf{ prop}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash B \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash A \oplus B \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash B \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash A \multimap B \textbf{ prop}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash {!A} \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash B \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash A \supset B \textbf{ prop}}$$

$$\frac{\Phi \vdash s : \textbf{sort} \quad \Phi; \Sigma, x{:}s; \Psi \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash \forall x{:}s.A \textbf{ prop}} \qquad \frac{\Phi \vdash s : \textbf{sort} \quad \Phi; \Sigma, x{:}s; \Psi \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash \exists x{:}s.A \textbf{ prop}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash K : \mathsf{principal} \quad \Phi; \Sigma; \Psi \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash \langle K \rangle A \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash I : \mathsf{interval}}{\Phi; \Sigma; \Psi \vdash A \mathbin{@} I \textbf{ prop}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash C \textbf{ constraint} \quad \Phi; \Sigma; \Psi, C \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash C \mathbin{\dot{\supset}} A \textbf{ prop}} \qquad \frac{\Phi; \Sigma; \Psi \vdash C \textbf{ constraint} \quad \Phi; \Sigma; \Psi, C \vdash A \textbf{ prop}}{\Phi; \Sigma; \Psi \vdash C \mathbin{\dot{\wedge}} A \textbf{ prop}}$$

$\boxed{\Phi; \Sigma; \Psi \vdash \gamma \textbf{ cat}}$

$$\frac{\Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash I : \mathsf{interval}}{\Phi; \Sigma; \Psi \vdash A[I] \textbf{ cat}}$$

$$\frac{\Phi; \Sigma; \Psi \vdash K : \mathsf{principal} \quad \Phi; \Sigma; \Psi \vdash A \textbf{ prop} \quad \Phi; \Sigma; \Psi \vdash I : \mathsf{interval}}{\Phi; \Sigma; \Psi \vdash (K \mathsf{\ affirms\ } A) \mathsf{\ at\ } I \textbf{ cat}}$$

Figure 5.4: The well-formedness rules for propositions and categorical judgments.

The judgment $\Phi; \Sigma; \Psi \vdash A$ **prop** means that A is a well-formed proposition. Its rules are given in Figure 5.4.

We should call particular attention to the rules for $C \mathbin{\dot{\supset}} A$ and $C \mathbin{\dot{\wedge}} A$. In both rules, proposition A is checked for well-formedness in the presence of constraint C, reminiscent of dependent product and sum types. The presence of constraint C is necessitated by the rule for interval well-formedness. Consider the proposition $\forall x{:}\mathsf{time}.(x \geq \bar{1}) \mathbin{\dot{\supset}} A \mathbin{@} [\bar{1}, x]$, for example. Without the constraint $x \geq \bar{1}$, the interval $[\bar{1}, x]$ (and consequently the whole proposition) is not well-formed. A similar example using $\dot{\wedge}$ is $\exists x{:}\mathsf{time}.(x \geq \bar{1}) \mathbin{\dot{\wedge}} A \mathbin{@} [\bar{1}, x]$.

In addition to propositions, categorical judgments are required. These correspond to the single-use truth and affirmation judgments from η_L logic:

$$\gamma ::= A[I] \mid (K \mathsf{\ affirms\ } A) \mathsf{\ at\ } I$$

We write $\Phi; \Sigma; \Psi \vdash \gamma$ **cat** for the judgment that γ is a well-formed categorical judgment. For it to be well-formed, each A, I, and K in γ must be a well-formed proposition, interval, and principal, respectively. Figure 5.4 gives the rules for this judgment.

5.1.5 Proof Terms and Their Typing Judgments

In order to verify the correctness of proofs through type checking, explicit proof objects, called proof terms, must be included in the formal system. Each proof term, denoted by the meta-variables M and N, corresponds to a single step in a derivation. We choose a set of natural deduction proof terms:

$$\begin{aligned}
M, N ::= {} & u \mid v \\
& \mid (M : \gamma) \\
& \mid M \otimes N \mid \mathbf{let}\ M = u_1 \otimes u_2\ \mathbf{in}\ N \\
& \mid \star \mid \mathbf{let}\ M = \star\ \mathbf{in}\ N \\
& \mid \langle M, N \rangle \mid \mathbf{fst}\ M \mid \mathbf{snd}\ M \\
& \mid \langle\rangle \\
& \mid \mathbf{inl}\ M \mid \mathbf{inr}\ M \mid (\mathbf{case}\ M\ \mathbf{of\ inl}\ u_1 \Rightarrow N_1 \mid \mathbf{inr}\ u_2 \Rightarrow N_2) \\
& \mid \hat{\lambda} i, u.M \mid M\hat{}N\ I \\
& \mid {!M} \mid \mathbf{let}\ M = {!v}\ \mathbf{in}\ N \\
& \mid \lambda i, v.M \mid M\ N\ I \\
& \mid \Lambda x.M \mid M[t] \\
& \mid \mathbf{pack}\ t\ \mathbf{with}\ M \mid \mathbf{let}\ M = \mathbf{pack}\ x\ \mathbf{with}\ u\ \mathbf{in}\ N \\
& \mid \_\ \mathsf{affirms}\ M \\
& \mid \langle \_ \rangle M \mid \mathbf{let}\ M = \langle \_ \rangle u\ \mathbf{in}\ N \\
& \mid\ @^{+} M \mid\ @^{-} M \\
& \mid \dot{\lambda}.M \mid M^{\cdot}\_ \\
& \mid \_ \mathbin{\dot{\wedge}} M \mid \mathbf{let}\ M = \_ \mathbin{\dot{\wedge}} u\ \mathbf{in}\ N
\end{aligned}$$

We use u and v (and their decorated variants) for linear and unrestricted variables, respectively. Because each variable can be identified as linear or unrestricted from the proof term that introduced it, this naming scheme carries no meaning, but is instead solely adopted for convenience. The syntax of the remaining proof terms is a mixture of notation from programming languages and notation mimicking the type corresponding to the proof term.

It should be noted that nearly all type annotations have been eliminated from the proof terms. (In some cases, the omitted annotations are replaced by _.) This is a consequence of the use of bidirectional typing judgments.

There are two bidirectional typing judgments for proof terms: a synthesis judgment $M \uparrow \gamma$ and a checking judgment $M \downarrow \gamma$.[1] As in earlier chapters, these basic judgments are extended to allow

[1]The reader may find it useful to note that the synthesis and checking judgments roughly correspond to neutral

$$\boxed{\Phi; \Sigma; \Psi \vdash \Gamma\ \mathbf{ok}}$$

$$\frac{\Phi; \Sigma \vdash \Psi\ \mathbf{ok}}{\Phi; \Sigma; \Psi \vdash \cdot\ \mathbf{ok}} \qquad \frac{\Phi; \Sigma; \Psi \vdash \Gamma\ \mathbf{ok} \quad v \notin \mathrm{Dom}(\Gamma) \quad \Phi; \Sigma; \Psi \vdash A\ \mathbf{prop} \quad \Phi; \Sigma; \Psi \vdash I : \mathsf{interval}}{\Phi; \Sigma; \Psi \vdash \Gamma, v{:}A[\![I]\!]\ \mathbf{ok}}$$

$$\boxed{\Phi; \Sigma; \Psi \vdash \Delta\ \mathbf{ok}}$$

$$\frac{\Phi; \Sigma \vdash \Psi\ \mathbf{ok}}{\Phi; \Sigma; \Psi \vdash \cdot\ \mathbf{ok}} \qquad \frac{\Phi; \Sigma; \Psi \vdash \Delta\ \mathbf{ok} \quad u \notin \mathrm{Dom}(\Delta) \quad \Phi; \Sigma; \Psi \vdash A\ \mathbf{prop} \quad \Phi; \Sigma; \Psi \vdash I : \mathsf{interval}}{\Phi; \Sigma; \Psi \vdash \Delta, u{:}A[I]\ \mathbf{ok}}$$

Figure 5.5: The well-formedness rules for proof contexts.

assumptions, which are ascriptions of types to proof variables in this case. The context Γ contains unrestricted variable typings, $v{:}A[\![I]\!]$, and Δ contains linear variable typings, $u{:}A[I]$. Figure 5.5 gives the well-formedness rules for these contexts. The full bidirectional typing judgment forms are:

$$\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \uparrow \gamma$$
$$\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \downarrow \gamma$$

These two judgments differ from an operational perspective. The type is considered an output of the synthesis judgment and an input to the checking judgment, justifying the names: $M \uparrow \gamma$ *synthesizes* a type γ for M, while $M \downarrow \gamma$ *checks* M at type γ. It is the operational distinction that allows most type annotations to be omitted from proof terms. In most cases where a type annotation would normally be needed, the checking judgment is used, and therefore the purported type is given as an input. This reduction of required type annotations is a standard benefit of bidirectional type checking [36].

5.1.6 Inference Rules

We now proceed to describe in detail a few inference rules for the typing judgments. As a general rule of thumb, introduction rules and closed-scope elimination rules (having proof terms of the form **let** $\ldots = \ldots$ **in** $\ldots$) use the checking judgment $M \downarrow \gamma$, while the remaining rules use the synthesis judgment $M \uparrow \gamma$.

First, compare the hyp rule for synthesizing a type for a linear hypothesis with the hyp′ rule for checking the type of a linear hypothesis:

$$\frac{}{\Phi; \Sigma; \Psi; \Gamma; u{:}A[I] \vdash u \uparrow A[I]}\ \mathsf{hyp} \qquad \frac{\Phi; \Sigma; \Psi \models I \supseteq I'}{\Phi; \Sigma; \Psi; \Gamma; u{:}A[I] \vdash u \downarrow A[I']}\ \mathsf{hyp}'$$

These rules reiterate the input-output mode difference between the synthesis and checking judgments. Because the hyp rule *synthesizes* a type for the hypothesis, it must output the largest correct

and normal deductions for logics. The directionality of the arrow notation, however, is often reversed in presentations of neutral and normal deductions.

interval; producing a smaller interval would unnecessarily throw away information. On the other hand, because the hyp' rule *checks* the type of the hypothesis against its input, any input interval I' smaller than the assumed interval I gives a valid type. (In practice, the hyp' rule is not needed: it is derivable from hyp and the other rules.)

Next, we examine the $\uparrow\downarrow$ rules for mediating between the two judgments:

$$\frac{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \uparrow A[I] \quad \Phi; \Sigma; \Psi \models I \supseteq I'}{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \downarrow A[I']} \uparrow\downarrow$$

$$\frac{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \uparrow (K \mathsf{~affirms~} A) \mathsf{~at~} I \quad \Phi; \Sigma; \Psi \models I \supseteq I'}{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \downarrow (K \mathsf{~affirms~} A) \mathsf{~at~} I'} \uparrow\downarrow\mathsf{affirms}$$

The $\uparrow\downarrow$ rule states that if type $A[I]$ can be synthesized for M and I' is a subinterval of I, then M checks against type $A[I']$. From an operational perspective, this means that if we are trying to check M against a type, we can synthesize a type for M and verify that the two types are related by subsumption. $\uparrow\downarrow\mathsf{affirms}$ is the analogous rule for affirmation types.

The $\downarrow\uparrow$ rule mediates between the two judgments in the opposite direction:

$$\frac{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash M \downarrow \gamma}{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash (M : \gamma) \uparrow \gamma} \downarrow\uparrow$$

This rule states that if the proof term M is annotated with a type γ and if M indeed checks against γ, then we may synthesize γ as the type of M.

Next, we consider the introduction and elimination rules for linear implication:

$$\frac{\Phi; \Sigma, i{:}\mathsf{interval}; \Psi, I \supseteq i; \Gamma; \Delta, u{:}A[i] \vdash M \downarrow B[i]}{\Phi; \Sigma; \Psi; \Gamma; \Delta \vdash \hat{\lambda} i, u.M \downarrow A \multimap B[I]} \multimap I$$

$$\frac{\Phi; \Sigma; \Psi; \Gamma; \Delta_1 \vdash M \uparrow A \multimap B[I] \quad \Phi; \Sigma; \Psi; \Gamma; \Delta_2 \vdash N \downarrow A[I'] \quad \Phi; \Sigma; \Psi \models I \supseteq I'}{\Phi; \Sigma; \Psi; \Gamma; \Delta_1, \Delta_2 \vdash M\hat{~}N\ I' \uparrow B[I']} \multimap E$$

The proof term for introduction of linear implication is $\hat{\lambda} i, u.M$, where i and u are the interval and proof assumptions created, respectively. (i is needed because intervals can appear in implication elimination proof terms.) Usually a lambda expression would have its parameter annotated with a type. However, because the $\multimap I$ rule checks against a type, writing $\hat{\lambda} i, u{:}A[i].M$ would be redundant: A is already known from the input to the checking judgment.

Most other type annotations on proof terms can be omitted for the same reason. This pattern does not apply to the elimination proof term for linear implication, however. The $\multimap E$ rule shows that a type must be synthesized for $M\hat{~}N\ I'$. By synthesizing a type for M, we learn A, B, and I. N must then be checked against $A[I']$. Without the annotation I' in the proof term, we would not know I' and would be forced to guess it. It is therefore not possible to maintain a standard bidirectional system and also eliminate the annotation I' from this proof term.

For the same reason, the elimination proof term for unrestricted implication also requires an interval annotation, while the introduction term needs no annotations.

The full set of inference rules for the typing judgments $M \uparrow \gamma$ and $M \downarrow \gamma$ is given in Figure 5.6. To avoid cluttering the rules, all contexts and types are assumed to be well-formed.

$$\dfrac{}{\Phi;\Sigma;\Psi;\Gamma;u{:}A[I]\vdash u\uparrow A[I]}\;\textsf{hyp}\qquad\qquad \dfrac{}{\Phi;\Sigma;\Psi;\Gamma,v{:}A\llbracket I\rrbracket;\cdot\vdash v\uparrow A[I]}\;\textsf{vhyp}$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\uparrow A[I]\quad \Phi;\Sigma;\Psi\models I\supseteq I'}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow A[I']}\;\uparrow\downarrow$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\uparrow(K\ \textsf{affirms}\ A)\ \textsf{at}\ I\quad \Phi;\Sigma;\Psi\models I\supseteq I'}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow(K\ \textsf{affirms}\ A)\ \textsf{at}\ I'}\;\uparrow\downarrow\textsf{affirms}$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow\gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash(M:\gamma)\uparrow\gamma}\;\downarrow\uparrow$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1\vdash M\downarrow A[I]\quad \Phi;\Sigma;\Psi;\Gamma;\Delta_2\vdash N\downarrow B[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2\vdash M\otimes N\downarrow A\otimes B[I]}\;\otimes I$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1\vdash M\uparrow A\otimes B[I]\quad \Phi;\Sigma;\Psi;\Gamma;\Delta_2,u_1{:}A[I],u_2{:}B[I]\vdash N\downarrow\gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2\vdash \textbf{let}\ M=u_1\otimes u_2\ \textbf{in}\ N\downarrow\gamma}\;\otimes E$$

$$\dfrac{}{\Phi;\Sigma;\Psi;\Gamma;\cdot\vdash\star\downarrow\mathbf{1}[I]}\;\mathbf{1}I\qquad\qquad \dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1\vdash M\uparrow\mathbf{1}[I]\quad \Phi;\Sigma;\Psi;\Gamma;\Delta_2\vdash N\downarrow\gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2\vdash\textbf{let}\ M=\star\ \textbf{in}\ N\downarrow\gamma}\;\mathbf{1}E$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow A[I]\quad \Phi;\Sigma;\Psi;\Gamma;\Delta\vdash N\downarrow B[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\langle M,N\rangle\downarrow A\,\&\,B[I]}\;\&I$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\uparrow A\,\&\,B[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\textbf{fst}\ M\uparrow A[I]}\;\&E_1\qquad\qquad \dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\uparrow A\,\&\,B[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\textbf{snd}\ M\uparrow B[I]}\;\&E_2$$

$$\dfrac{}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\langle\rangle\downarrow\top[I]}\;\top I$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\textbf{inl}\ M\downarrow A\oplus B[I]}\;\oplus I_1\qquad\qquad \dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash M\downarrow B[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\textbf{inr}\ M\downarrow A\oplus B[I]}\;\oplus I_2$$

$$\dfrac{\begin{array}{c}\Phi;\Sigma;\Psi;\Gamma;\Delta_1\vdash M\uparrow A\oplus B[I]\\ \Phi;\Sigma;\Psi;\Gamma;\Delta_2,u_1{:}A[I]\vdash N_1\downarrow\gamma\\ \Phi;\Sigma;\Psi;\Gamma;\Delta_2,u_2{:}B[I]\vdash N_2\downarrow\gamma\end{array}}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2\vdash\textbf{case}\ M\ \textbf{of}\ \textbf{inl}\ u_1\Rightarrow N_1\mid\textbf{inr}\ u_2\Rightarrow N_2\downarrow\gamma}\;\oplus E$$

$$\dfrac{\Phi;\Sigma,i{:}\textsf{interval};\Psi,I\supseteq i;\Gamma;\Delta,u{:}A[i]\vdash M\downarrow B[i]}{\Phi;\Sigma;\Psi;\Gamma;\Delta\vdash\hat{\lambda}i,u.M\downarrow A\multimap B[I]}\;\multimap I$$

$$\dfrac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1\vdash M\uparrow A\multimap B[I]\quad \Phi;\Sigma;\Psi;\Gamma;\Delta_2\vdash N\downarrow A[I']\quad \Phi;\Sigma;\Psi\models I\supseteq I'}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2\vdash M\hat{}N\ I'\uparrow B[I']}\;\multimap E$$

Figure 5.6: The bidirectional typing rules.

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\cdot \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\cdot \vdash !M \downarrow !A[I]}\ !I \qquad \frac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1 \vdash M \uparrow !A[I] \quad \Phi;\Sigma;\Psi;\Gamma, v{:}A\llbracket I\rrbracket;\Delta_2 \vdash N \downarrow \gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2 \vdash \textbf{let}\ M = !v\ \textbf{in}\ N \downarrow \gamma}\ !E$$

$$\frac{\Phi;\Sigma, i{:}\mathsf{interval};\Psi, I \supseteq i;\Gamma, v{:}A\llbracket i\rrbracket;\Delta \vdash M \downarrow B[i]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \lambda i, v.M \downarrow A \supset B[I]}\ {\supset}I$$

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \uparrow A \supset B[I] \quad \Phi;\Sigma;\Psi;\Gamma;\cdot \vdash N \downarrow A[I'] \quad \Phi;\Sigma;\Psi \models I \supseteq I'}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M\ N\ I' \uparrow B[I']}\ {\supset}E$$

$$\frac{\Phi;\Sigma, x{:}s;\Psi;\Gamma;\Delta \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \Lambda x.M \downarrow \forall x{:}s.A[I]}\ \forall I \qquad \frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \uparrow \forall x{:}s.A[I] \quad \Phi;\Sigma;\Psi \vdash t : s}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M[t] \uparrow [t/x]A[I]}\ \forall E$$

$$\frac{\Phi;\Sigma;\Psi \vdash t : s \quad \Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \downarrow [t/x]A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \textbf{pack}\ t\ \textbf{with}\ M \downarrow \exists x{:}s.A[I]}\ \exists I$$

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1 \vdash M \uparrow \exists x{:}s.A[I] \quad \Phi;\Sigma, x{:}s;\Psi;\Gamma;\Delta_2, u{:}A[I] \vdash N \downarrow \gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2 \vdash \textbf{let}\ M = \textbf{pack}\ x\ \textbf{with}\ u\ \textbf{in}\ N \downarrow \gamma}\ \exists E$$

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \_\ \mathsf{affirms}\ M \downarrow (K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I}\ \mathsf{affirms} \qquad \frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \downarrow (K\ \mathsf{affirms}\ A)\ \mathsf{at}\ I}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \langle\_\rangle M \downarrow \langle K\rangle A[I]}\ \langle\rangle I$$

$$\frac{\begin{array}{c}\Phi;\Sigma;\Psi;\Gamma;\Delta_1 \vdash M \uparrow \langle K\rangle A[I] \\ \Phi;\Sigma;\Psi \models I \supseteq I' \\ \Phi;\Sigma;\Psi;\Gamma;\Delta_2, u{:}A[I] \vdash N \downarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'\end{array}}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2 \vdash \textbf{let}\ M = \langle\_\rangle u\ \textbf{in}\ N \downarrow (K\ \mathsf{affirms}\ B)\ \mathsf{at}\ I'}\ \langle\rangle E$$

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash @^{+}M \downarrow A\ @\ I[I']}\ @I \qquad \frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \uparrow A\ @\ I[I']}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash @^{-}M \uparrow A[I]}\ @E$$

$$\frac{\Phi;\Sigma;\Psi, C;\Gamma;\Delta \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \dot{\lambda}.M \downarrow C \mathrel{\dot{\supset}} A[I]}\ \dot{\supset}I \qquad \frac{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \uparrow C \mathrel{\dot{\supset}} A[I] \quad \Phi;\Sigma;\Psi \models C}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M^{\cdot}\_ \uparrow A[I]}\ \dot{\supset}E$$

$$\frac{\Phi;\Sigma;\Psi \models C \quad \Phi;\Sigma;\Psi;\Gamma;\Delta \vdash M \downarrow A[I]}{\Phi;\Sigma;\Psi;\Gamma;\Delta \vdash \_ \mathbin{\dot{\wedge}} M \downarrow C \mathbin{\dot{\wedge}} A[I]}\ \dot{\wedge}I$$

$$\frac{\Phi;\Sigma;\Psi;\Gamma;\Delta_1 \vdash M \uparrow C \mathbin{\dot{\wedge}} A[I] \quad \Phi;\Sigma;\Psi, C;\Gamma;\Delta_2, u{:}A[I] \vdash N \downarrow \gamma}{\Phi;\Sigma;\Psi;\Gamma;\Delta_1,\Delta_2 \vdash \textbf{let}\ M = \_ \mathbin{\dot{\wedge}} u\ \textbf{in}\ N \downarrow \gamma}\ \dot{\wedge}E$$

Figure 5.7: The bidirectional typing rules, continued.

5.2 Implementing the Proof Checker

The combination of linearity and constraints in η_L logic introduces several, albeit small, implementation challenges. The techniques used in resolving these problems are not unique to this implementation, but are instead borrowed from previous work. We first discuss explicit substitutions and de Bruijn indices as general techniques, and then turn our attention to linearity and the constraint solver.

5.2.1 Explicit Substitutions

To avoid implementing a direct substitution function for terms and proofs, one can encode the substitution operation as an explicit proof term. Substitutions can then be lazily computed during type checking. This method also prevents the size explosion that results from directly substituting a large object for many variable occurrences.

Despite the advantages of explicit substitutions, there is a possible tradeoff. With the addition of substitution objects, any hope for the equivalence of terms and propositions based on purely syntactic means is lost. For example, the term t and the substitution $[t/x]$ applied lazily to x are equivalent, but do not share the same syntax. This burdens the implementation with functions for normalizing terms and propositions and for determining equivalence of these normal forms.

In the end, explicit lazy, and not direct eager, substitutions based on the $\lambda\sigma$ calculus of Abadi *et al.* [5] were implemented in the proof checker. Because neither the advantage of efficiency nor the disadvantage of additional code were particularly compelling factors for a small proof-of-concept implementation, the use of explicit substitutions was primarily chosen as an exercise.

5.2.2 de Bruijn Indices

Rather than using a named representation for proof and term variables, de Bruijn indices were chosen. de Bruijn indices name each occurrence of a variable according to the number of variable bindings that separate that occurrence from its binder. For example, $\lambda x.\lambda y.y\ (\lambda z.x\ z)$ would be represented with de Bruijn indices as $\lambda.\lambda.1\ (\lambda.3\ 1)$.

de Bruijn indices simplify the implementation of a type checker by eliminating the need for α-conversion; unlike named representations of proof terms, α-equivalent terms have syntactically identical de Bruijn representations since the underlying names are ignored. In addition, they cooperate well with explicit substitutions. It is for these reasons that de Bruijn indices were chosen.

It is natural to expect that the implementation would mirror the formal system by separating the assumptions into the five contexts. However, this interacts poorly with de Bruijn indices. Suppose that, as part of an atomic proposition, some type in Δ mentions the term parameter with de Bruijn index 1 in Σ. Now, if a new term parameter is bound, the index 1 in this type must be shifted to 2, or else the type will refer to the wrong parameter. Such shifts would need to occur for Ψ, Γ, and Δ every time a new term parameter is bound.

A better solution is to combine all of the five contexts into one. Under the convention that a context entry is well-formed in the tail of its context, shifts will not be incurred for each new term parameter. Instead, each de Bruijn index in the context will remain fixed thoughout its lifetime.

Only when a type or constraint is used from the context will it need to be shifted. This is the approach taken by the proof checker implementation.

5.2.3 Linearity

Enforcing the single-use nature of linear hypotheses during type checking requires careful consideration. At a first glance, the problem might appear relatively straightforward: type checking should simply ensure that each hypothesis is distributed to exactly one recursive call. But, the situation is complicated by the nondeterministic presentation of some of the inference rules. As an example, consider the $\otimes I$ rule:

$$\frac{\Phi; \Sigma; \Psi; \Gamma; \Delta_1 \vdash M \downarrow A[I] \quad \Phi; \Sigma; \Psi; \Gamma; \Delta_2 \vdash N \downarrow B[I]}{\Phi; \Sigma; \Psi; \Gamma; \Delta_1, \Delta_2 \vdash M \otimes N \downarrow A \otimes B[I]} \otimes I$$

Δ_1, Δ_2 is the multiset of resources that must be used in the proof term $M \otimes N$. However, there are exponentially many ways to distribute these resources among the premises, and it would seem difficult to find a correct distribution. In fact, through the variables it uses, the proof term lists (almost) all of the resources consumed. The input-output method [15] leverages this information to eliminate the nondeterminism of splitting the context.

Under this approach, the proof term $M \otimes N$, for example, is checked against $A \otimes B[I]$ as follows. First, *all* of the incoming resources, say Δ^+, are used to check M against $A[I]$. By mentioning only some of the variables from Δ^+, the proof term M consumes only some of these resources and outputs the rest, say Δ. Then, Δ is used to check N against $B[I]$. Again, N consumes only some of these resources and outputs the rest, say Δ^-. Thus, M greedily (and deterministically) decided how to divide the resources between itself and N.

The implementation of the proof checker follows this model. However, explicitly returning only the unconsumed part of the context would interfere with the numbering scheme for de Bruijn indices: the de Bruijn index would no longer correspond to the distance from the front of the context. So, instead, the *full* context is returned after flagging each resource as "consumed" or "unconsumed."

This marking scheme is further complicated by $\langle\rangle$, the proof term for $\top$, because it can consume resources but does not list them explicitly. If Δ^+ is used to check $\langle\rangle$ against $\top[I]$, then $\langle\rangle$ might consume these resources, or it might produce them as output. Therefore, a third flag is introduced: "possibly consumed." Because it can consume any unconsumed resources in the current context, $\langle\rangle$ represents a fallback option for any of those resources that are not later consumed. $\langle\rangle$ therefore re-marks all "unconsumed" resources as "possibly consumed." Since $\langle\rangle$ is only a fallback, later proof terms may re-mark "possibly consumed" resources as "consumed."

The implementation, by using this three flag input-output method, can deterministically and efficiently type check linear proof terms.

5.2.4 Constraints

The superset constraint solver for the proof checker posed a unique challenge: how should superset constraints reconcile interval parameters i and explicitly constructed intervals $[t, t']$?

This can be resolved by requiring all terms of sort $\mathsf{interval}$ to be explicitly constructed intervals. Under this approach, the superset constraint $[t_1, t_2] \supseteq [t'_1, t'_2]$ is considered an abbreviation for a conjunction of two, now primitive, $\geq$ constraints among times: $t'_1 \geq t_1$ and $t_2 \geq t'_2$. The $\geq$ constraints can then be solved using a simple decision procedure that builds a reflexive, transitive closure by saturation.

For this to work, only the representation of interval parameters needs to be changed. Consider the introduction of an interval parameter i by universal quantification: $\forall i{:}\mathsf{interval}.A$. Instead of using i directly, two time parameters i_1 and i_2, with $i_2 \geq i_1$, are created and i is replaced with $[i_1, i_2]$. Then, checking can continue. Thus, the original proposition is effectively translated to:

$$\forall i_1{:}\mathsf{time}.\forall i_2{:}\mathsf{time}.(i_2 \geq i_1) \mathbin{\dot{\supset}} ([[i_1, i_2]/i]A)$$

Similar operations are performed for the corresponding proof term and for the introduction of interval parameters in the existential quantification and implication rules. Because interval parameters are eliminated immediately before use, no other cases need to be considered.

5.3 Conclusion

In this chapter, we have described a proof checker for η_L logic. We first reduced the level of abstraction by refining the treatment of sorts and atomic propositions, and then examined the intricacies of verifying the well-formedness of terms, particularly intervals, and propositions. Next, we presented a bidirectonal type checking system upon which the proof checker is built. Finally, we discussed a few of the implementation challenges and the techniques used to resolve them.

Chapter 6

Conclusion

In this thesis, we have argued that, to be widely applicable, authorization logics should be able to express the kinds of time-dependent access control policies that arise naturally in practice. This thesis has therefore focused on the development and study of an authorization logic with an explicit notion of time.

In summary, this thesis makes two conceptual contributions and a small practical contribution. First, by developing η_N logic, it demonstrates that an authorization logic suitable for expressing time-dependent policies can be easily obtained by relativizing the judgments of a time-unaware authorization logic to time intervals. Moreover, through meta-theoretic properties, this thesis has shown the logic to be sound and formally proven it to be an extension of previous work.

Second, by extending η_N logic with linearity to create η_L logic, this thesis has exhibited the ability of linearity and explicit time to coexist in a logic. Also, a careful study of the logic's meta-theory was carried out.

Finally, by presenting a bidirectional type checker, this thesis has made the first, very small step toward a full-scale PCA architecture based on η_L logic. The successful implementation of this type checker suggests that such a PCA system should be easily constructible.

Despite these contributions, significant opportunities for continued research on η logic remain. This thesis is therefore concluded with a brief description of possible future work.

6.1 Future Work

Directions for future research relating to η logic encompass both immediate and long-term goals. We outline these possibilities in order of increasing scope.

Formal Comparison to Other Logics and Languages. The authorization logics and languages that handle time of which we are aware all do so by extra-logical mechanisms. In contrast, η logic internalizes time. To transport results from other logics to η logic, it would be useful to formally compare η logic with these differing approaches. In particular, the $\mathsf{currentTime}()$ predicate of SecPAL [10] would serve as an interesting starting point.

PCA Architecture. A stated goal of this thesis was the design of a logic for time-dependent authorization policies that could serve as the foundation for a PCA-based policy enforcement mechanism. Since its design has now been completed, it is natural to consider creating and deploying a full-scale PCA architecture based on η logic. In fact, this work is already underway; as a component of his doctoral thesis, Deepak Garg is implementing a PCA file system for his own variant of η logic [22].

Policy Analysis. As the access control policies of systems are often exceedingly numerous and complex, policy analysis tools are critical if authorization logics are to be successfully adopted in practice. Such analysis can be carried out for the logic as a whole using non-interference theorems, and for specific policies using the completeness of focusing (for example).

The study of policy analysis for an authorization logic via non-interference theorems was initiated by Garg and Pfenning [24]. For example, they established an affirmation flow analysis theorem for GP logic: in the absence of a connection (or flow of affirmation) between principals K_1 and K_2, no statement made by K_1 can influence K_2's affirmations. Abadi subsequently showed a related property for DCC [2]. We expect that these results can be adapted to η logic. Additionally, it will likely be possible to prove other non-interference theorems specific to time, including a "time flow" analysis: in the absence of a connection between intervals I_1 and I_2, no event occurring during I_1 can influence an event during I_2.

In related work, Garg *et al.* [23] also developed a method, based on the completeness of focusing, for establishing the correctness of specific policies. We believe that η logic satisfies the completeness of focusing, but it needs to be formally verified. After doing so, it should be possible to analyze individual policies by this method.

Privacy Policies. Complementing access control, privacy policies represent the other crucial component of security. The growing emphasis placed on privacy is evidenced by the large number of recent regulations enacted by the United States government, such as the Health Insurance Portability and Accountability Act (HIPAA) and the Family Educational Rights and Privacy Act (FERPA).

With its importance to security, efforts toward a combined logic for authorization and privacy are underway. Garg *et al.* [23] have introduced $K \;\mathsf{has}\; A$, a modality for modeling possession. We believe that the addition of a time-dependent has to η logic would permit certain time-dependent privacy policies to be expressed. For example, K might be allowed to circumvent a given privacy policy, provided he can satisfy an obligation A during interval I. It seems plausible that the obligation could be represented as something like $(K \;\mathsf{has}\; (A \multimap \mathbf{1}))[I]$. Further examination of practical privacy policies is needed to verify this claim.

Bibliography

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

[2] Martín Abadi. Access control in a core calculus of dependency. In *Proceedings of the 2006 ACM International Conference on Functional Programming (ICFP '06)*, pages 263–273, New York, New York, 2006. ACM Press.

[3] Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '99)*, pages 147–160, San Antonio, Texas, January 1999.

[4] 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, September 1993.

[5] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, October 1991.

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

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

[8] Lujo Bauer, Scott Garriss, Jonathan M. McCune, Michael K. Reiter, Jason Rouse, and Peter Rutenbar. Device-enabled authorization in the Grey system. In *Information Security: 8th International Conference, ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 431–445, September 2005.

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

[10] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. Design and semantics of a decentralized authorization language. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF '07)*, pages 3–15, Venice, Italy, July 2007. IEEE Computer Society Press.

[11] Patrick Blackburn. Representation, reasoning, and relational structures: A hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[12] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael 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.

[13] Torben Braüner and Valeria de Paiva. Towards constructive hybrid logic. In *Electronic Proceedings of Methods for Modalities*, September 2003.

[14] Jan G. Cederquist, Ricardo Corin, Marnix A. C. Dekker, Sandro Etalle, Jerry I. den Hartog, and Gabriele Lenzini. Audit-based compliance control. *International Journal of Information Security*, 6(2):133–151, 2007.

[15] Iliano Cervesato, Joshua S. Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. *Theoretical Computer Science*, 232:133–163, 2000.

[16] Rohit Chadha, Damiano Macedonio, and Vladimiro Sassone. A hybrid intuitionistic logic: Semantics and decidability. *Journal of Logic and Computation*, 16(1):27–59, 2006.

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

[18] John DeTreville. Binder, a logic-based security language. In Martín Abadi and Steven Bellovin, editors, *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 105–113, Berkeley, California, May 2002. IEEE Computer Society Press.

[19] Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF-21)*, Pittsburgh, Pennsylvania, June 2008. IEEE Computer Society Press. To appear.

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

[21] Thom Frühwirth. Temporal annotated constraint logic programming. *Journal of Symbolic Computation*, 22:555–583, 1996.

[22] Deepak Garg. Logic-based authorization. PhD thesis proposal, March 2008.

[23] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of affirmation and knowledge. In Dieter Gollman, Jan Meier, and Andrei Sabelfeld, editors, *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS '06)*, pages 297–312, Hamburg, Germany, September 2006.

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

[25] 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.

[26] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[27] Limin Jia. *Linear Logic and Imperative Programming.* PhD thesis, Department of Computer Science, Princeton University, November 2007.

[28] Max I. Kanovich, Mitsuhiro Okada, and Andre 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, 1998.

[29] 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, November 1992.

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

[31] Davide Marchignoli. *Natural Deduction Systems for Temporal Logics.* PhD thesis, University of Pisa, February 2002.

[32] 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.

[33] Ben Moszkowski and Zohar Manna. Reasoning in interval temporal logic. In Edmund Clarke and Dexter 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] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001.

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

[36] Benjamin C. Pierce and David N. Turner. Local type inference. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '98)*, San Diego, California, 1998.

[37] Jason Reed. Hybridizing a logical framework. In *Proceedings of the International Workshop on Hybrid Logic (HyLo 2006)*, volume 174 of *Electronic Notes in Theoretical Computer Science*, pages 135–148, 2007.

[38] Uluç Saranlı and Frank Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems. In *Proceedings of International Conference on Robotics and Automation (ICRA '07)*, Rome, Italy, April 2007. IEEE Computer Society Press.

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