Proof Theory for Authorization Logic and Its Application to a Practical File System

Deepak Garg

CMU-CS-09-168
December 2009

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Frank Pfenning, Chair
Martín Abadi
Lujo Bauer
Anupam Datta
Robert Harper

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

This research was supported partially by the Air Force Research Laboratory under grant no. FA87500720028, and partially by the iCAST project sponsored by the National Science Council, Taiwan under grant no. NSC97-2745-P-001-001.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Proof Theory for Authorization Logic and Its Application to a Practical File System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University ,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**302**	

Keywords: Proof theory, access control, file system, formal logic, modal logic, computer security

Abstract

In most computer systems, users' access to resources is controlled using authorization policies. Logic is an appropriate medium for representing, understanding, and enforcing authorization policies, yet despite several years of pragmatic work on the subject, the foundations of relevant logics remain unexplored and poorly understood. It is in this realm that the work of this thesis lies; the thesis explores the theory of logics for expressing authorization policies as well as applications of the theory in practice. In doing so, it makes three foundational and technically challenging contributions.

First, the thesis introduces *proof theory* and metatheory in the context of authorization logics, illustrated through a new logic BL. In particular, structural proof-theoretic systems of natural deduction and sequent calculus are investigated and their importance explained. Pragmatic problems like proof verification and automatic proof search are then addressed using the sound foundations of proof theory.

Second, the thesis considers a logical treatment of *dynamism* in authorization policies and, in particular, logical constructs for representing authorizations depending on system state, consumable credentials, and explicit time are presented. Further, a practical, efficient, and provably correct mechanism for their enforcement is developed. The mechanism is based on a combination of proofs and cryptographic capabilities.

Third, the *practical usefulness* of the proof theory and the enforcement mechanism is demonstrated through an implementation of the same in a file system, PCFS. It is shown through measurements that file access in PCFS is very efficient.

In addition, the thesis includes a detailed *case study* that formalizes in BL policies used to control access to classified information in the U.S., and explains how the policies may be enforced using PCFS.

To integrity and trust, whose lapses motivated this thesis

Acknowledgments

Numerous people have contributed directly or indirectly to the development of this thesis and I would like to express my gratitude to each of them. Frank Pfenning, my advisor, has been a continuous source of inspiration and support, and a very patient and helpful mentor for the past six years. I cannot thank him enough for making my life as a Ph.D. student the pleasure it has been. Likewise, Anupam Datta has been extremely supportive and helpful in the past two years. Martín Abadi, Lujo Bauer, and Bob Harper have all inspired parts of this thesis and without their encouragement this thesis would have remained incomplete. Earlier, several faculty members at IIT-Delhi introduced me to the fascinating world of research. I wish to thank all these individuals for their continued encouragement and feedback.

My family, both immediate and distant, have always supported me and I extend my heartfelt gratitude to all of them. For the same reason, I am also grateful to all my friends in Pittsburgh and outside. Finally, I wish to extend my warmest thanks to members of Mount Carmel School as well as my parents for laying the foundations of my present knowledge.

Contents

Chapter 1

Background and Motivation

1.1 Background: The Problem of Access Control

Both for the purpose of security and as good programming practice, the access that principals (users, programs, etc.) have to resources is often restricted. This practice, generically called access control, is pervasive; its use ranges from low level memory subsystems where programs are limited to reading and writing their own memory pages to applications like web servers where web documents are protected from unauthorized users. Despite differences in both the resources protected and principals from whom they are protected, the high level architecture of most access control mechanisms is similar – all calls that access a protected resource pass through a subsystem called the *reference monitor*, which, based on the identity of the principal making the call, the resource being accessed, and the nature of the call (read, write, create, etc.), either allows the call to proceed or blocks it. The process by which the reference monitor identifies the principal making the call is called *authentication*, whereas the process of deciding whether to allow access or not is called *authorization*. Authentication, although very important, is a well-studied problem with solutions that work in almost any setting. For example, one may use passwords or secret keys for authentication. This thesis focuses on the other problem – authorization. We use the term "authorization policy" or policy to refer to rules on which authorization decisions are based.

Authorization. A significant question in the design of an access control subsystem is how the reference monitor decides which requests to authorize and which to deny. One possibility, which is unfortunately often used, is to encode this information in the program of the reference monitor, making no separation between the code and the authorization policy. This approach is non-modular and requires that code be changed every time the policy changes. The other possibility is to make the policy an input to the code, perhaps to be read from a configuration file. Separating code from policy makes the access control subsystem both more robust and modular. In particular, changes to the policy, including the special case of fixing errors in it, do not require changes to code. Further, the code and the policy can be analyzed separately for correctness.

Access control lists and their problems. Given that policy and code should be separated, the next important question is how the policy may be represented, and how consequences may be drawn from it. A common solution to this problem is to model the policy as a table that for every principal k, resource r, and operation o tells whether principal k may perform operation o on resource r or not [89]. This model of the access control policy is called an access control matrix. An access control matrix is often represented by storing its entries with the respective resources to which they apply. The entries stored with each resource are called an access control list (ACL). Although access control lists are both simple to implement and very widely used in practice, they are low level representations of the policy and suffer from the drawback that they do not carry information about *why* a certain access is allowed or denied. As a result, in scenarios where accountability of access is a concern (e.g., we would like to know why an individual was able to read a file, not merely that she was able to), additional mechanisms must be provided to record and track reasons for entries in access control lists. Among other applications, such accountability is important in military and intelligence servers with classified information, for businesses that have proprietary data to protect, and in matters of customer privacy. The second problem with access control lists is that it is difficult to keep them up to date with changing access requirements, and this often results in policy errors and inadvertent accesses. We illustrate the limitations of access control lists in the following example.

Example 1.1. Consider a hypothetical scenario where Alice is an employee of the company AuthCo, and within the company works for the team GovTeam, which handles contracts from the government. As a member of the team Alice has access to a government dataset d. Owing to the sensitive nature of the dataset, this access is contingent upon her maintaining a government security clearance.

Suppose that the access control policy for the dataset d is represented using ACLs. While Alice has access, her name would be on the ACL of d. Observe, however, that the access control list does not provide any evidence as to why Alice has this access (it does not record her affiliation with GovTeam, nor her security clearance). As a result if an internal auditor were to try to determine whether it is legitimate to have Alice's name on the ACL or not, he would have to consult many other sources. Further, if Alice were to lose her government security clearance, some administrator would have to manually observe this change and go ahead and remove her name from the ACL. If for any reason the administrator failed to take notice, Alice would continue to have access when she should not, resulting in a security breach.

Rule-based representation. The problems with ACLs, as illustrated by Example 1.1, can be eliminated using a rule-based representation of policies. Intuitively, in such a representation the policy is represented as a set of if-then rules, and access is allowed only if it is entailed by the rules. As an illustration, the policy in Example 1.1 may be expressed by the following rules.

1. For any principal k, if k works for GovTeam and k has a government security clearance then k can read dataset d.

2. Alice works for GovTeam.

3. Alice has a government security clearance.

The main advantage of representing the policy as rules is that the reason for access becomes explicit. Here for instance, were an audit to be performed, it would be clear that Alice has access because she works for GovTeam and also has a government security clearance. Further, when Alice wants to read d, the reference monitor (or, as we shall see later, Alice) must *infer* that these three rules entail that Alice may do so, and this inference can be logged as evidence that explains *why* Alice obtained access. This increases accountability and improves assurance in the access control subsystem. The second advantage of using rules is that such a representation can be implemented to propagate the policy change automatically with conditions. (This is explained in §1.2.2.) For example, if Alice were to lose her government security clearance, there would no longer be any inference to authorize Alice's read request, and hence she would no longer be able to read d.

The role of formal logic. The next relevant issue is determining a formal language that may be used to represent policy rules and determine their consequences. While there are many different kinds of existing formal languages for representing policy rules, this thesis rests on the idea that logic may be used to represent policy rules and to enforce them (see §3.7 for a description of some other formalisms). This is an observation that goes back to Lampson and others [88]. As an example, the policy rules (1)–(3) may be represented by the three formulas below, assuming that the predicate $\texttt{worksFor}(k, \mathsf{GovTeam})$ means that k works for GovTeam, $\texttt{hasClearance}(k)$ means that k has government security clearance, and $\texttt{may}(k, d, \mathsf{read})$ means that k is allowed to read dataset d.

1'. $\forall k.\ ((\texttt{worksFor}(k, \mathsf{GovTeam}) \wedge \texttt{hasClearance}(k)) \supset \texttt{may}(k, d, \mathsf{read}))$

2'. $\texttt{worksFor}(\mathsf{Alice}, \mathsf{GovTeam})$

3'. $\texttt{hasClearance}(\mathsf{Alice})$

The reader may check that in either classical or intuitionistic logic, (1')–(3') entail the formula $\texttt{may}(\mathsf{Alice}, d, \mathsf{read})$. This idea of using logic to represent policy rules and to find their consequences is the starting point for this thesis, and hence extremely important from our perspective. The following are some reasons to show that the use of logic for these purposes is not a mere convenience, but, in fact, very natural and pragmatic.

- Once represented in logic, the consequences of the policy rules are unambiguous since they are defined by the logic's semantics. Hence, logic provides a rigorous foundation for *defining* the meanings of policies.

- A logical proof that shows why policy rules authorize access may be used to gain access, and it may also be logged to improve the *accountability* of the access control subsystem.

- Logical inference and automatic proof search based on it can be used to *implement* the policy rules directly. This point is elaborated in the rest of this thesis.

1.2 Technical Background

Having justified the importance of logic in the context of authorization, we now turn to technical work in the area which serves as background for the thesis. First, we describe some existing work on logics that are well suited for representing policies (called authorization logics) and, second, we describe proof-carrying authorization, a formal mechanism for enforcement of policies represented in logic. These two together also lead directly to the motivation for this thesis (§1.3).

1.2.1 Authorization Logics

Although many authorization policies may be represented in propositional or first-order logic as, for example, we did in §1.1, there are some commonly occurring policy idioms that are best represented with specialized logical connectives (see §3.1.2 for examples). Many logics with such specialized connectives have been proposed, e.g., [5, 8, 13, 18, 54, 65–67, 88]. We use the term *authorization logic* to designate any logic that has been designed with the explicit purpose of representing authorization policies.

In addition to authorization logics, there is also a significant amount of past work on logic-based declarative languages for writing authorization policies and determining their consequences, e.g, [23, 26, 49, 52, 118]. Most of these languages have a syntax that resembles the syntax of logical formulas, and their inference rules are often based on logic programming.

Although we postpone a comparison of different authorization logics and logic-based authorization languages to later chapters (§3, §4, and §9), we discuss here, in brief, one connective that is common to many authorization logics and authorization languages. This connective, written k says s, was first introduced in an authorization logic by Lampson et al. [8, 88]. k says s means that principal k says, claims, or supports the formula s, but does not imply that s is true. The connective is useful for representing authority of principals on parts of policies, and for capturing the interactions between rules created by different principals, as the following example illustrates.

Example 1.2. Continuing the scenario of Example 1.1, let us assume that there are three principals involved in authorization: (a) admin who has ultimate authority on deciding who should have access to the dataset d, (b) AuthCoHr which determines team affiliations of employees of AuthCo, and (c) Gov which determines government security clearances. Accordingly, the policy rules (1)–(3) from §1.1 may be refined by the rules shown below to specify these authorities. The principal who certifies (creates) each rule is indicated at the beginning of the rule in square brackets [·].

1a. [admin] For every principal k, if AuthCoHr certifies that k works for GovTeam and Gov certifies that k has a security clearance, then k is allowed to read dataset d.

2a. [AuthCoHr] Alice works for GovTeam.

3a. [Gov] Alice has a government security clearance.

Using the connective k says s, these rules may be represented as follows.

1a'. admin says $\forall k.$ (((AuthCoHr says $\texttt{worksFor}(k, \text{GovTeam})) \wedge (\text{Gov says } \texttt{hasClearance}(k))) \supset \texttt{may}(k, d, \text{read})$)

2a'. AuthCoHr says $\texttt{worksFor}(\text{Alice}, \text{GovTeam})$

3a'. Gov says $\texttt{hasClearance}(\text{Alice})$

In general, principal Alice will be allowed to read dataset d only if there is a proof of the following formula: admin says $\texttt{may}(\text{Alice}, d, \text{read})$. In most authorization logics, but not all, (1a')–(3a') entail this formula.

1.2.2 Proof-carrying Authorization (PCA)

Proof-carrying authorization (PCA), earlier called proof-carrying authentication, is a rigorous mechanism based in cryptography and formal proofs that is used for *distributed enforcement* of authorization policies represented in logic. It was introduced by Appel and Felten [13], and has since been used for access control both on the web [18] and in physical devices like office doors [20], as well as in language interfaces for controlling access to sensitive resources like files [15, 41, 85]. Although PCA has traditionally relied on policies represented in higher-order logic, its central ideas (listed below) generalize to any authorization logic.

- Principal k' should be allowed to perform operation o on resource r if and only if k' can produce a *formal proof* which shows that the policy rules in effect entail that access should be allowed. For instance, in the example of §1.1, k' would have to prove the formula admin says $\texttt{may}(k', r, o)$ from the policy rules (1a')–(3a').

- Policy rules may be established using digital signatures: if principal k signs the formula s with its private key, then the resulting digital certificate is evidence that k says s holds.[1] (k says s can also be inferred from other formulas via the logic's inference system.)

Based on these ideas, PCA allows distributed enforcement of authorization policies represented in logic in the following manner. Administrators sign policy rules in digital certificates which are then published through any mechanism (such as a LDAP server). A principal k' desirous of access selects certificates it believes relevant to authorizing its access, and taking the policy rules instated by the certificates as hypothesis, constructs a logical proof M which establishes that it has legitimate access. Along with its request to perform the access, k' also provides the proof M and the certificates used in it to the reference monitor (hence the adjective "proof-carrying"). The reference monitor verifies the digital certificates by checking the digital signatures in them, and also verifies the logical proof M. If both checks succeed, the access is allowed, else it is blocked.

[1] In a lot of work on PCA [13, 18, 20], a digital certificate establishes a formula k signed s, which implies but is not implied by, k says s. However, in this thesis we blur the distinction, since we have no occasion to use k signed s.

Since the principal requesting access must provide the certificates on which the proof relies, the reference monitor is freed from the responsibility of tracking all policy rules in effect. This makes PCA truly distributed. The main reason that PCA requires that the principal requesting access provide a proof authorizing its access, as opposed to the reference monitor finding the proof itself via automatic proof search, is one of efficiency. It is a well-known fact that for most logics proof verification is straightforward and takes time linear in the size of the proof, whereas the time for proof search is at least exponential in the size of the hypotheses and the formula being established; in most cases proof search is undecidable. By distributing the work of inference to principals, PCA not only prevents proof search from making the reference monitor a performance bottleneck, but also admits the possibility of allowing each principal to use other information such as its knowledge of context and state, as well as human intervention to quickly construct proofs.

Whereas the PCA architecture works well in settings like web services where communication delay overshadows verification time significantly, for low level interfaces like file systems, even proof verification at each access becomes a performance bottleneck (this is explained in Section 1.3). As a result in these situations, PCA is not appropriate, and one of the significant contributions of this thesis is a rigorous architecture for policy enforcement that overcomes this problem (§5), and a practical demonstration in a file system that it is actually efficient (§7).

1.3 Motivation for the Thesis

There are three main motivations for the work in this thesis, which we explain in this section.

Motivation 1 (Proof-theoretic foundations)**.** As should be obvious from the description of proof-carrying authorization, logical inference and proofs play a significant role in enforcement of policies represented in logic. Yet, surprisingly, prior to the joint work of the author and Pfenning [67], which in a sense forms the foundation for the theoretical ideas in this thesis, there was hardly any systematic investigation of proof theory of authorization logics. Authorization logics up to that point were either described axiomatically or via inference rules that had little justification besides the fact that they suited the intended applications. Structural proof systems such as natural deduction and the sequent calculus [70] were missing, as was a description of metatheoretic properties of authorization logics, such as admissibility of cut and consistency. Besides their use in proof-carrying authorization, good proof theory and metatheoretic properties are important in the context of authorization for the following reasons:

- Allowed inferences or proofs define the meanings of authorization policies and, as a result, semantics other than proof-theoretic (model-theoretic, Kripke, set-theoretic) are of secondary importance for authorization.

- Proof theory can be used to justify the foundations of the logic in the form of metatheoretic properties like cut elimination and symbolic consistency, as in a lot of prior work on other logics [39, 99, 115].

- Proof-theoretic results can be used directly in practical tools such as provers and verifiers, both of which are essential in logic-based enforcement of authorization.

A thorough investigation of proof theory and metatheory of a specific authorization logic constitutes a significant portion of the theoretical work in this thesis and forms the foundation for the rest of the thesis. It should be emphasized here that owing to specialized constructs like the modality k says s and many others that we introduce in this thesis, proof theory of authorization logics is non-trivial and proof-theoretic results from classical and intuitionistic logics do not directly apply.

Motivation 2 (Support for dynamic policies)**.** As was illustrated briefly in Example 1.1, allowed accesses in practice are not static, but change with time. Being able to express such possibly changing (dynamic) policies in an authorization logic and enforcing them with proof-carrying authorization is a significant challenge, and forms the next motivation for this thesis. In general, dynamism in policies can be of different types, some of which we summarize below.

- *Start and expiration*: A policy rule may come into effect at a stipulated point of time. Similarly, it may expire at a stipulated point of time. Representing either of these requires that there be an explicit representation of clock time in the logic.

- *State dependence*: An authorization may be allowed only while the system is in a certain state, which may change unpredictably. Representing state dependence requires a syntax to mark predicates as being external to the logic, and a formal incorporation of system state in the proof system.

- *Consumption*: A permission may be usable a finite number of times. Representing such permissions requires that the logic be able to count resources.

- *Revocation*: A policy rule may be revoked by its creator at a time that is not predictable, e.g., because the rule was created in error. Although revocation cannot be represented in a logic (a priori, in a logic the hypotheses are assumed to hold), it is nonetheless important in practice.

Prior to the work of the author, often jointly with others [54, 66], there was no systematic mechanism to represent any of these forms of policy dynamism in authorization logics. Logic-based authorization languages included limited support for some of these but the solutions lacked a logical foundation (see §4.7 and §9.4 for a discussion of some of this work). In this thesis, we present systematic logical ways to express expiration, state dependence, and consumption in authorization logic, and describe how all of these as well as revocation may be enforced in an extension of proof-carrying authorization.

Motivation 3 (Efficiency of enforcement)**.** Proof-carrying authorization (§1.2.2) is practical in only those scenarios where proof verification can be performed fast enough to not make the reference monitor a bottleneck in practice. Although the exact time varies, proof verification in practice takes several milliseconds at the least. Most of this time is spent

in reading proofs and certificates from storage, and in parsing them. Whereas this time frame is acceptable in many access control scenarios such as network services and in physical devices where other processes like network communication and movement of mechanical parts take much longer, it is slow enough to make the reference monitor a bottleneck in operation-intensive applications like file systems.[2] Making proof-carrying authorization efficient enough for use in operation-intensive settings like file systems, without losing any of its rigorous guarantees forms the motivation for the implementation work in this thesis. We argue in this thesis that PCA can be complemented with capabilities to attain this efficiency, without losing any of its formal rigor, and demonstrate the practicality of the architecture through a file system implementation.

Thesis Statement. Based on the motivations described above, the statement of the thesis is:

"Logic, grounded in strong *proof theory*, can be used for representing and reasoning about *dynamic* authorization policies, and for *efficiently* enforcing them."

1.4 Summary of Work in the Thesis

The work in this thesis can be divided roughly into three parts: (a) Theoretical work on the proof theory of a new authorization logic called BL, (b) The design and implementation of a practical file system, PCFS, that relies on the logic's proof theory and metatheoretic properties for representation of authorization policies and their enforcement, and (c) A case study of real policies for access control on classified information in the U.S. to demonstrate the usability of both BL and PCFS.

The logic BL. The presentation of the authorization logic BL focuses on its proof theory and metatheoretic properties for reasons mentioned in §1.3 (Motivation 1). We present both a natural deduction proof system, which provides an intuitive proof-theoretic explanation of the meanings of connectives and forms the basis of proofs, and a sequent calculus, which is useful for proof search and as a tool for proving theorems about the logic.

The choice of a new logic, as opposed to the possibility of using an existing authorization logic, is justified because BL is better suited to representing policies than existing authorization logics. In particular, BL is intuitionistic, first-order, and interprets the modality k says s in a novel way. The use of intuitionistic logic, as opposed to prior proposals which were classical, is explained at the end of this section. First-order quantification is important because it arises naturally in policies that often are generic in principals, resources, etc. Interestingly, all authorization logics prior to the work of the author [67] were either propositional or higher-order, although some logic-based authorization languages allowed first-order quantification, e.g., [52]. The unique interpretation of k says s in BL permits the representation of certain forms of delegation that are important in practice; this is justified in §3.1.2. We formally establish the expressiveness of BL through sound and complete

[2] To confirm this hypothesis, we implemented a file system that verifies proofs at each access, and noticed visible delays even in simple operations like listing directory contents in a shell.

embeddings from an existing authorization logic, and an existing logic-based authorization language into a fragment of BL (§3.5).

Another important emphasis in the design of BL is representation and enforcement of dynamic policies, as described in §1.3 (Motivation 2). The logic BL contains explicit time, as well support for predicates that are external to the logic and interpreted directly on the state of the system (§4). The treatment of explicit time is based on joint work with DeYoung and Pfenning [54], and builds on ideas from DeYoung's undergraduate thesis [53]. Predicates interpreted on the state of the system are novel to this thesis. An extension of BL, called BL^{L} uses ideas from linear logic [71] and builds on prior joint work of the author [66] to allow for representation of certificates that can be used a finite number of times only (§9).

In addition to proof theory and metatheoretic properties of BL, the thesis also considers their practical implications in the form of procedures for proof verification and proof search. A detailed investigation of proof terms and their practical verification (which is complicated due to the presence of explicit time and state) is considered in §5. A practical method for proof search that builds on the proof theory and literature on logic programming is presented in §6.

The file system PCFS. As observed in §1.3 (Motivation 3), the PCA idea of verifying proofs during each call to the reference monitor is infeasible in heavily used settings like file systems because reading and parsing of proofs and certificates makes the reference monitor a bottleneck. Consequently, this thesis proposes and demonstrates the implementation of a revised architecture where the work of proof verification is offlined to trusted verifier(s) outside the reference monitor. The trusted verifiers issue very simple cryptographic capabilities in return, which may be presented to the reference monitor as evidence of authorized access. These capabilities, called procaps, can be verified in a few microseconds each, which allows the reference monitor to handle thousands of calls a second. While the idea of offlining policy enforcement is not new, and may seem very trivial, what makes the design difficult and technically challenging is the interaction between dynamic policy elements, proofs and procaps. If, for example, a proof relies on a policy rule that is set to expire at a stipulated time, this constraint on time must be reflected in any procap generated from the proof. The procaps, therefore, are *conditional* on those policy elements in proofs that are dynamic.

We show in this thesis that any constraints on proofs due to explicit time and dependence on system state can be systematically extracted in the proof verification procedure and written to capabilities; we also show rigorously that the extracted constraints suffice to establish that the original proof still holds at a later point of time, which immediately implies that the enhanced architecture is equivalent (in terms of allowed accesses) to PCA (§5). It therefore achieves high efficiency without comprising any of PCA's rigor. As a practical demonstration we have implemented a file system, PCFS, that uses the architecture and through experimental measurements we show that the file system allows high throughput of up to several thousand file system calls a second (§7). We also discuss how the other two forms of policy dynamism listed in §1.3, namely revocation and consumption, may be implemented through procaps.

At a high-level of abstraction, procaps used in PCFS are similar to entries of a cache that a reference monitor may maintain to record accesses it has authorized in the past. However, procaps are more general than cache entries since they scale easily to decentralized settings where the trusted verifier and reference monitor are running on different nodes of a network. Further, as opposed to a reference monitor maintained cache, procaps are closer in spirit to proof-carrying authorization where the principal seeking access is responsible for maintaining and providing evidence to authorize access. Another merit of using capabilities as opposed to caches is that both the design and implementation of the access control system factor into two parts that interact via capabilities only: (a) The *front end* that deals with policies, proofs, and verification of proofs and certificates, and (b) The *back end* that uses capabilities to authorize access and perform I/O. Indeed, the PCFS back end is independent of the logic used in the front end, and it can be used with any policy infrastructure that produces compatible capabilities.

A salient feature of PCFS, unrelated to logic, is its backwards compatibility with existing programs and nearly complete POSIX compliance. This compatibility is the result of two design decisions. First, instead of requiring that procaps be passed during file system calls (which would require a change to the system call API), procaps are stored in a central location which PCFS looks up automatically. Second, files created by a program remain accessible to it temporarily via default procaps that are generated by the file system itself. This allows programs to create and use temporary files without having to generate proofs. As a result, PCFS is able to run most existing applications including word processors and spreadsheets without any changes to the applications themselves, which makes the file system practical.

Case study. To verify and demonstrate the expressiveness of BL as a logic for representing authorization policies, and of PCFS as a file system for enforcing them, a large case study of policies that are used to control the dissemination of classified information in the U.S. is presented in §8. The policies used in the case study are based on actual data obtained from the U.S. intelligence community by Symantec Corporation, and given to Carnegie Mellon University as part of a joint government contract. The case study is extensive and uses most features of BL including its support for explicit time and system state.

On the Use of Intuitionistic Logic in the Thesis

Following prior work with Pfenning [67], the logic BL is intuitionistic. In contrast, older authorization logics were classical [8, 18, 20, 88]. The move from classical logic to intuitionistic logic in the context of authorization is based on the fact that intuitionistic logic requires constructive evidence for formulas whereas classical logic does not. For instance, in intuitionistic logic, a closed proof of $s_1 \vee s_2$ always either contains a closed proof of s_1 or a proof of s_2 (modulo proof normalization). This is not the case in classical logic wherein $s_1 \vee s_2$ can be established by showing that the simultaneous falsity of s_1 and s_2 would entail a contradiction. Constructive evidence has important consequences for accountability in authorization. For instance, consider the following two policy rules p_1 and p_2, both of

which allow access to an office door (predicate $\texttt{mayenter}(x)$). p_1 allows employees access on weekdays, and p_2 allows managers access on other days, i.e. weekends.

$$\begin{array}{rcl} p_1 & : & \forall x.\ ((\texttt{weekday} \wedge \texttt{employee}(x)) \supset \texttt{mayenter}(x)) \\ p_2 & : & \forall x.\ (((\neg\texttt{weekday}) \wedge \texttt{manager}(x)) \supset \texttt{mayenter}(x)) \end{array}$$

Suppose that Alice is a manager and an employee, and let e and m be proofs that establish these to be the case.

$$\begin{array}{rcl} e & : & \texttt{employee}(\mathsf{Alice}) \\ m & : & \texttt{manager}(\mathsf{Alice}) \end{array}$$

Assuming that formulas are interpreted classically, let *dis* stand for an instantiation of the law of the excluded middle to the formula $(\texttt{weekday} \vee (\neg\texttt{weekday}))$. Then, the following is a proof of $\texttt{mayenter}(\mathsf{Alice})$.

$$\begin{array}{l} \texttt{case}\ (dis)\ \texttt{of} \\ \quad \texttt{inl}\ v \Rightarrow p_1\ \mathsf{Alice}\ \langle v, e\rangle \\ \mid \texttt{inr}\ w \Rightarrow p_2\ \mathsf{Alice}\ \langle w, m\rangle \end{array}$$

This proof simply case analyzes the disjuncts in $(\texttt{weekday} \vee (\neg\texttt{weekday}))$; in one case it uses p_1, in the other it uses p_2. The important observation here is that although the proof establishes access for Alice, it does not make the reason for access explicit. More specifically, the proof does not state whether Alice has access because it is a weekday and she is an employee, or because it is a weekend and she is a manager. Clearly, if such a proof were to be used to audit any specific access that Alice performed, it would provide little insight.

If intuitionistic logic were to be used instead of classical logic, then this proof would be disallowed since *dis* would not be valid. Instead, Alice would be *forced* to provide evidence of either $\texttt{weekday}$, or of $\neg\texttt{weekday}$. In either case, the proof would contain the reason for access explicitly. It is for such increased accountability that intuitionistic logic has been chosen for the work in this thesis.

1.5 Contributions of the Thesis

This thesis makes three main contributions to the area of authorization logics and their implementation, as well as several minor ones.

Main contributions. The three main contributions of the thesis are:

- Investigation of proof theory and metatheoretic properties for authorization logics, illustrated through a new logic BL
- Logical treatment of dynamism in authorization policies; in particular, dependence on state and consumable credentials, and a practical, efficient, and provably correct mechanism for enforcement of policies dependent on state as well as explicit time
- An enhanced PCA-based architecture for enforcement of policies expressed in logic that is efficient enough for use in operation-intensive systems, and its practical demonstration through an implementation in a file system, PCFS

Minor contributions. In addition, the thesis makes several minor contributions, many of which are technical in nature.

- The new, expressive authorization logic BL and a formal justification of expressiveness via translations from existing policy frameworks
- A systematic, proof-theoretic description of a proof search procedure for an authorization logic, grounded in existing work on logic programming
- Detailed investigation of properties of proofs in authorization logic and their verification
- A detailed case study of policies for access to classified information in the U.S. and their formalization
- An implementation of a PCA-based file system that is backwards compatible and largely POSIX compliant
- On a more foundational level, a logic (BL) whose hypothetical judgments are indexed by first-order terms, and where truth is always relativized to principals

1.6 Aspects of Authorization Not Covered in the Thesis

Although the broad theme of this thesis is authorization logics and their applications, certain aspects of the use of authorization logic are not covered in the thesis. In order to correctly qualify the scope of the thesis, we list these out-of-scope topics below.

- Policy administration: The thesis does not cover issues of how responsibilities of administering different parts of policies are distributed to individuals, nor how they may change. Instead, in all examples, we assume that these are provided a priori. Other literature on policy administration deals with these aspects, e.g., [93, 126, 135].
- Policy authoring: Tools for writing policies in logical form are not covered in the thesis. In practice, such tools are necessary because policy administrators cannot be expected to understand formal logic.
- Policy storage and distribution: We do not describe possible ways in which policy certificates may be stored or distributed. In particular, for proof search (§6), we ignore the issue of finding relevant certificates that may be needed in the proof. This problem has been studied extensively in the past, e.g., [21, 24, 46].
- Correctness of policies with respect to intent of the creator: The thesis does not consider the problem of whether a policy represented in authorization logic has the meaning its author intended. This is a question of policy analysis, and although logical techniques may be used to assist in addressing the problem [5, 42, 67], there is an element of subjectivity in it, as described in a study on differences in human intent and actual representation of access policies [19].

- Authentication: We implicitly assume that there is a mechanism for the reference monitor to identify the principal making an access request. For PCFS, which runs as a kernel service in Linux, authentication is quite trivial since the POSIX method `getuid`() is used to identify the user making a file system call.

1.7 Outline of the Thesis

The rest of this thesis is organized as follows.

In §2, we provide a brief overview of the PCFS architecture, and describe how its various components work with each other in practice. This provides a perspective for the work in the rest of the thesis.

§3 introduces a fragment of BL called BL_S that contains only k says s in addition to the usual connectives of first-order logic. After a discussion of an axiomatic proof system and some examples, proof theory (natural deduction and sequent calculus) and metatheoretic properties like admissibility of cut are introduced. The nature of the says modality in BL_S is elaborated through translation to a previously known modal logic, and the expressiveness of BL_S is established by translating two known policy formalisms into it.

§4 generalizes the work in §3 to the full logic BL that includes explicit time and predicates interpreted on system state, thus paving the way for representing dynamic policies. Again, proof theory and metatheory are investigated, and in addition, proof normalization is studied. The chapter also discusses how BL is used in PCFS.

In §5, proof terms for BL are studied and a practical, bidirectional procedure for their verification is presented, which is implemented in PCFS. One novel contribution of this chapter is the method for extracting dynamic policy elements from a proof, and a proof that these elements, together with verification performed by trusted proof verifiers, suffice to authorize access correctly. This shows that the PCFS architecture achieves the same security guarantees as PCA. The chapter also describes the structure of procaps and how they are verified, as well enforcement of policy revocation.

§6 describes the method of proof search used in the prover included in PCFS. Building on ideas from existing work in logic programming, the chapter identifies a very expressive fragment of BL on which goal-directed search is complete, and proves that this is the case.

§7 describes the implementation of PCFS in detail. In particular, it discusses how the theory in §5 and §6 is used in practice. It also describes the back end of the file system that includes the layout of files, directories, and the protection of configuration files. Performance measurements establishing the high efficiency attained by PCFS are also presented in this chapter.

§8 presents a case study on the use of BL and PCFS. The case study considers real policies for access control on classified information in the U.S.

§9 considers an extension of BL with ideas from linear logic, which allows representation of credentials that can be used a fixed number of times only. Proof theory and metatheory of this extension, BL^{L}, are presented, as are some examples of its use. In addition, the chapter describes a method for enforcing consumable credentials in the PCFS architecture.

§10 concludes the thesis with some directions for future work.

Work most closely related to each chapter is presented at its end. Readers interested in understanding only PCFS but not the proof theory of BL are advised to read §2, §4.1, §4.3, §5.2, and §7.

Chapter 2

An Overview of the Proof-Carrying File System (PCFS)

This chapter provides a brief overview of the architecture of PCFS, the file system that is designed and implemented in this thesis. The purpose of presenting the architecture upfront is to provide a perspective for both the theoretical work on BL (§3, §4) and the description of proof verification and proof search (§5, §6), as well as to highlight the overall merits of the architecture. Details of the design, implementation, and evaluation of PCFS are postponed to §7.

PCFS builds on ideas from proof-carrying authorization (§1.2.2). It is currently implemented as a local file system for the Linux operating system, but its architecture has been designed to support distribution. The name PCFS is an acronym for Proof-Carrying File System, even though access requests in PCFS *do not* carry proofs as they do in proof-carrying authorization. Instead, proof verification is delegated to offline trusted verifiers that are invoked prior to file access.

Briefly, PCFS works as follows. The access policy is represented as logical formulas in BL and distributed to users in the form of digital certificates signed by policy administrators. A user constructs formal proofs, which show that the policy entails certain permissions for her. Each proof is checked by a trusted proof verifier which gives the user a signed capability in return. This capability, called a *procap* (for proven capability), can be used repeatedly to authorize access to file system operations; the file system checks the procap each time it is required for authorization. Therefore, policy enforcement in PCFS follows the path:

$$\text{Policy} \rightarrow \text{Proof} \rightarrow \text{Procap} \rightarrow \text{File access}$$

2.1 The PCFS Architecture

Figure 2.1 shows the PCFS architecture. Numbers are used to label steps in order in which they occur in practice. Steps 1–6 deal with the logic, and include proof generation, proof verification, and creation of procaps. These steps are performed in advance of file access, and happen infrequently (usually when a user accesses a file for the first time). Once procaps

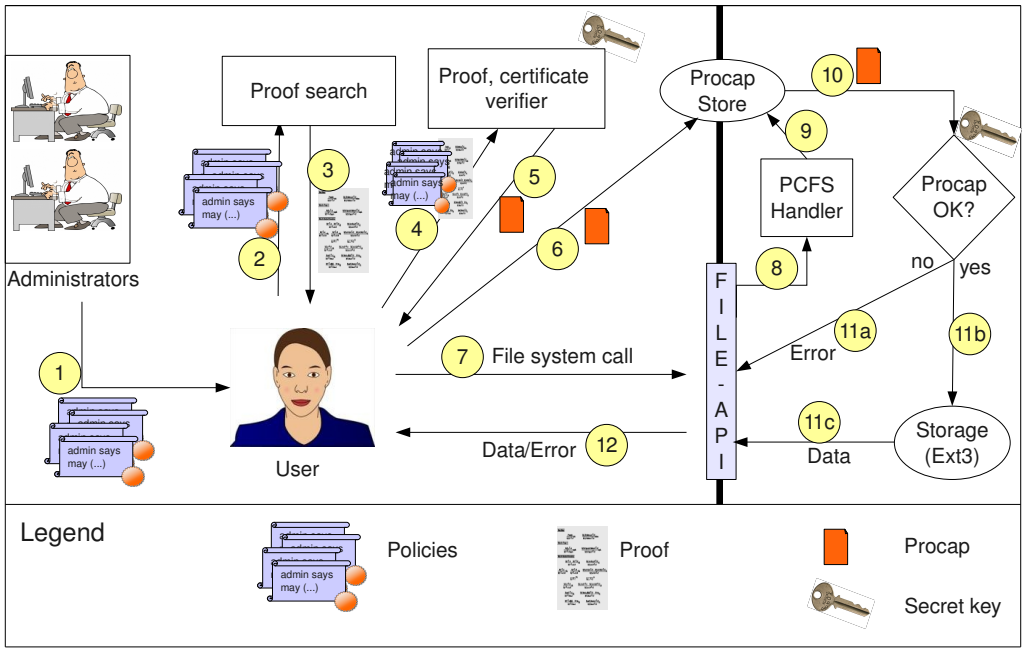

Figure 2.1: PCFS architecture

are stored, they can be used repeatedly to perform file operations (steps 7–12). The solid black vertical line in the diagram separates parts that happen in user space, i.e. before and after a file system call (left side of the line) from those that happen during a file system call (right side of the line). In the following we describe the steps of Figure 2.1 in some detail.

Policy creation (Step 1). A policy is defined as a set of formulas in the logic BL (§4) that determine access rights. An access right is a triple $\langle k, f, \eta \rangle$, which means that user k (Alice, Bob, etc) has permission η (read, write, etc) on file or directory f. A policy is concretely represented as digital certificates, signed by individuals who create it. PCFS provides a command line tool, `pcfs-cert`, to help administrators check formulas for adherence to logical syntax, to digitally sign them, and to convert them to a custom certificate format. (We could have used a standard certificate format like X.509 [79], but found it easier to create our own format.)

Proof generation (Steps 2–3). Once certificates have been created by administrators and given to users, the latter use them to show that they are allowed certain permissions in the file system. The basic tenet of PCFS, as in PCA, is that a user k is allowed permission η on resource f at time u, if and only if the user can provide a *formal logical proof* M, which shows that the policy in effect, Γ, entails a fixed formula $\texttt{auth}(k, f, \eta, u)$ or, in formal notation, $\Gamma \vdash M :: \texttt{auth}(k, f, \eta, u)$. The formula $\texttt{auth}(k, f, \eta, u)$ (actually a logical

judgment) is defined in §4.3.

To help users construct the proof M, PCFS provides an automatic theorem prover for BL, through the command line tool `pcfs-search`. This tool is based in logic programming; its underlying theory is the subject of §6. Figure 2.1 shows the user giving the policy (certificates) to the proof search tool in step 2, and the proof search tool returning a proof in step 3. A typical proof construction in PCFS takes several hundred milliseconds. A salient point is that the proof search tool is *not* a trusted component of PCFS and a user may use any method to create proofs.

Proof verification (Steps 4–5). Once the user has constructed a proof M, this proof, together with the certificates used to construct it, is given to a proof verifier, invoked using another command line program `pcfs-verify` (Step 4 in Figure 2.1). The verifier is a trusted component of PCFS. It checks that the logical structure of the proof M is correct, and that all certificates used in the proof are genuine, i.e. their digital signatures check correctly. If both these hold, then the verifier gives back to the user a procap, which is a capability that mentions the right $\langle k, f, \eta \rangle$ that the proof grants (Step 5). The procap also contains some conditions on which the proof depends and is signed using a shared symmetric key that is known only to the verifier and the file system interface (see §7 for details). The method used for verification of BL proofs and extraction of conditions from them is discussed in §5. A typical proof verification including creation of a procap takes several tens or a few hundred milliseconds, depending on the size of the proof.

Procap injection (Step 6). After receiving a procap, the user invokes another command line tool to put the procap in a central store, marked "Procap Store" in Figure 2.1. This store is in a designated part of the PCFS file system, and is accessible to both users and the system interface. The system interface looks up this store to find relevant procaps when file system calls are made. The organization of the store is described in §7.

File system call (Step 7). A call to the PCFS file system is made through the usual POSIX file system API during the execution of a user program. PCFS respects the standard POSIX interface, so user programs and shell commands don't need to change to work on it. However, before a file system call is executed the user or the program must ensure that procaps to authorize the call have been created and injected using Steps 2–6.

Procap look up and checking (Steps 8–10). Once a program has made a file system call the file system looks up one or more procaps to authorize the operation (Steps 9 and 10). Procaps needed to authorize common operations are listed in §7. If all relevant procaps are found, they are checked. Checking a typical procap takes only 10–100μs (*cf.* the time taken to check a proof, which is of the order of tens or hundreds of milliseconds). Details of procap checking are presented in §5.

Error reporting (Steps 11a, 12). If any procap needed for performing the requested file operation is missing or fails to check an error code is returned to the user program.

File operation (Steps 11b, 11c, 12). If all relevant procaps needed to perform the requested file operation are found, and successfully check, then the file operation is performed. In the current implementation of PCFS, actual I/O is performed by redirecting to an existing file system (Step 11b). Hence PCFS is a *virtual file system* that layers logic-based access control on another file system.

2.2 Comparison to Proof-Carrying Authorization

The architecture of PCFS extends proof-carrying authorization (PCA) [13] with procaps. The PCFS architecture and PCA differ in at least two ways. First, in PCA, a proof authorizing access is verified during each call to a resource, whereas in PCFS proofs are verified in advance of access and exchanged for procaps that authorize system calls. This allows much higher throughput at the resources (files) because checking procaps is faster than checking proofs. Experimental measurements of the performance of PCFS are presented in §7.

The second difference between the PCFS architecture and PCA is more an artifact of the manner in which the latter has been implemented in many systems, rather than a fundamental distinction. As it turns out, many PCA implementations [18, 20] use a challenge response protocol during access, as part of which the principal requesting access is given a nonce. This nonce must be embedded in the proof used to authorize access because the interface does not learn the identity of the principal. This implies that the proof cannot be completed in advance of the access (although most parts of the proof are independent of the nonce and can be constructed in advance). The PCFS architecture, on the other hand, necessitates that the entire proof be constructed and verified in advance of access and that the reference monitor learn the identity of the principal making the file system call, because that identity is matched to the identity listed in the procap used for authorization.

In addition to these differences, the logics used as the basis of many PCA implementations differ significantly from BL. These differences are discussed in §3.7 and §4.7.

2.3 Merits of the PCFS Architecture

Interestingly, besides the obvious merit of improving throughput in access to resources, the PCFS architecture has two other significant merits.

Modularity. Owing to the separation of the proof verifier from the reference monitor, the access control subsystem factors into two parts, both conceptually and in the implementation: (a) the *front end*, which understands the logic and digital certificates, and performs proof search and proof verification to generate procaps, and (b) the *back end*, which checks procaps to authorize access and performs resource access. The two parts only interact through procaps and are otherwise independent. In Figure 2.1, the front end corresponds to steps 1–6 and the back end corresponds to steps 7–12. This factorization has the following merits:

- The front end may be changed to support a different logic, or even replicated to support two authorization logics simultaneously, without any need to change the back end.

- The same front end can be used with different back ends.

- The front end and back end can be implemented, tested, and debugged separately, possibly by different teams having expertise in logic and systems programming respectively. For example, in the current implementation of PCFS, the front end is written in SML, while the back end is written in C++ and has been optimized for speed. There are no compile time dependencies between the two parts. However, both parts agree on a common structure for procaps.

Backwards compatibility. By storing procaps in a central location ("procap store" in Figure 2.1) rather than requiring programs to provide them at the time of access, as PCA does for proofs, PCFS is able to maintain backwards compatibility with the POSIX file system interface. This allows existing programs to run without modification, provided that enough procaps are generated in advance to authorize all access they need. A complication arises for files that programs create while they execute, in particular, temporary files that word processors and spreadsheets often create. To allow programs to access such files without the need to create and check proofs, the file system automatically generates default procaps that give the creating user read and write access to a new file or directory for a certain period of time. As a result, even sophisticated software like word processors and spreadsheets work seamlessly on PCFS. Access through default procaps can be turned off by changing an extended attribute on the file or directory on which such access is conditional.

2.4 Related Work

Proof-Carrying Authorization. As noted earlier, proof-carrying authorization was first described by Appel and Felten [13]. There are currently two large implementations of PCA [18, 20]. The more recent of these, called Grey, is a generic architecture for access control that is currently deployed for access to office doors in one floor at Carnegie Mellon University. More recently, Lesniewski-Laas et al. [90] have described an extension of PCA in which credentials define not only authorization policies but also the cryptographic primitives and credential formats that may be used to incorporate policies from external systems.

Vaughan et al. [139] describe an architecture similar to PCA, focusing on a "proofs as log entries" approach, where full proofs used for access are written in logs. Their architecture is based on a strongly typed language, and it is assumed that proofs passed to the reference monitor are correct (so there is no need for proof-checking), but proofs may be inspected subsequently for audit. In joint work, the author and Chaudhuri describe a compiler that supports PCA like interfaces (in particular, PCFS) by automatically inserting code to generate and verify proofs [41]. Avijit et al. [15] and, independently, Vaughan et al. [85] present programming languages whose resource access APIs are guarded by proof-carrying autho-

rization. In both these languages, the type system ensures that correct proofs are presented at each access.

Logic-based authorization in Taos. Prior to the advent of proof-carrying authorization, Wobber et al. designed, implemented and tested logic-based authentication and authorization for the distributed operating system Taos [143]. In their design, a logic is used to *authenticate* the caller to the callee in a remote procedure call (RPC): through logical inference, the callee learns the identity of the remote principal p on behalf of whom the channel c over which the call comes is acting. The relation between c and p is expressed as the logical formula $c \Rightarrow p$, read "c speaks for p" (the logic used is that of Lampson, Abadi and others [8, 88]).

Although PCFS does not directly use this idea, the work is closely related to PCFS for two reasons. First, the callee of an RPC in Taos may request the caller to provide evidence which establishes $c \Rightarrow p$. This evidence, which is encoded as an S-expression, is similar to a logical proof and contains signed certificates at its leaves. The callee verifies the evidence, and in the process learns p. In this sense, the work on Taos is a precursor to proof-carrying authorization since the latter generalizes the idea of using proofs for authentication to using proofs for authorization. Second, efficient performance of the authentication mechanism in Taos relies on caching of proofs and, further, cached proofs expire automatically when the certificates embedded in them expire. In this sense too, the work is related to PCFS: as mentioned in §1, procaps in PCFS generalize the idea of a cache of authorizations by allowing for distribution, and enforce not only time-based expiration but also state-dependent invalidation of proofs.

Trust Management and digital certificates. The idea of using digital certificates to represent policies, although not in logical form, dates back at least to the description of X.509 certificates [79]. Prior to its adoption in PCA, the idea evolved in many other policy frameworks, including PolicyMaker [33], KeyNote [31], and SPKI [58]. To the best of our knowledge, the use of digital certificates to establish policies in logical form was first considered in work on the Taos operating system discussed above.

Authorization in file systems. POSIX standards for access control in file systems [134] follow the UNIX model [137] where read, write, and execute permissions for the owner and the owning group of a file or directory are stored in file system meta-data. File systems that use this model include early versions of NFS [125], SFS [100], Truffles [124], and most file systems for UNIX-like environments. Access control in other file systems, including AFS [130], NTFS [48], NFSv4 [133], CIFS [77], GSFS [87], and most file systems on newer Linux kernels, relies on access control lists that allow or deny permissions to all users (not only the owner). Although the exact number and kinds of permissions vary, this model is an improvement over the UNIX model since it allows administrators to give access to arbitrary users, without having to add them to the file's owning group. In the file system Bayou [136], authorization is based on certificates that are signed by a single trusted administrator.

A significant limitation of both the UNIX model and per-user ACLs is that there is no easy way to allow an ordinary user to give permissions to other users. (As explained in §3.1.2, this kind of delegation is straightforward in PCFS.) In some file systems like AFS, users may be given administrative rights over the ACL of a file system object, through which they may add other users to the object's ACL. However, there is no way to limit this authority, say, to specific permissions. In file systems such as Truffles, Bayou, and WebFS [138] users may transfer their permissions on files to other users by signing certificates.

File systems with authorization mechanisms closest to those of PCFS are Echo (the file system in Taos discussed earlier), DisCFS [104], WebDAVA [91], and Fileteller [81]. The last three of these use the Trust Management system KeyNote [31] for authorizing access. Like proof-carrying authorization and PCFS, trust management frameworks admit flexible policies that are represented in digitally signed certificates. However, as opposed to proof-carrying authorization and PCFS, inference from certificates is performed by the reference monitor, which may cause problems at large scales. A survey article by Miltchev et al. [105] reviews and compares authorization mechanisms in networked file systems.

Many existing file systems, including CapaFS [123] and several file systems for network-attached storage disks [10, 73, 112, 121], use capabilities to authorize access. However, capabilities in these file systems differ from those in PCFS significantly. In these file systems, capabilities are sufficient to authorize access. On the other hand, capabilities in PCFS are only a tool for improving efficiency and backwards compatibility – access is still contingent on proofs; capabilities only carry information about proof verification and dynamic constraints in proofs from the proof verifier to the reference monitor (§5).

Chapter 3

BL_S: An Authorization Logic for Static Policies

This chapter describes an authorization logic BL_S, which is suitable for expressing static authorization policies. Static authorization policies, as opposed to dynamic authorization policies, do not rely on time and state. §4 describes a larger logic BL that, unlike BL_S, contains support for explicit time, constraints, and predicates interpreted on the state of the system. These features can be used to express dynamic authorization policies.

BL_S is an extension of first-order intuitionistic logic with a single modality k says s, which means that principal k says, claims, or supports the truth of formula s but does not imply that s is true. In practice, the modality is used to distinguish policy rules and credentials created by different individuals. For example, if principal k signs a certificate containing formula s, this may be reflected in the logic as the formula k says s. Whereas the idea of using a modality of this nature to distinguish policies of different principals is not new, and goes back to the work of Lampson et al. from 1992 [88], the proof-theoretic interpretation of the modality k says s in BL_S is original.

The purpose of considering BL_S separately from the full logic BL is to make the presentation easier to follow. First, working with a simple logic like BL_S makes it easier to introduce basic concepts of structural proof theory such as the sequent calculus and natural deduction, as well as their metatheoretic properties like admissibility of cut in the context of authorization. These form the centerpiece of the rest of this thesis. Second, this chapter considers translations from two existing formalisms for expressing authorization policies, namely, the GP logic [67] and Soutei [118], to BL_S and compares a third formalism, Binder [52], to BL_S. For these purposes, there is no need to consider the full logic BL because, like many other authorization formalisms, GP logic, Soutei, and Binder do not consider explicit time or state. Third, this chapter connects BL_S to the modal logic constructive S4 [11, 115], and a fragment of BL_S to intuitionistic first-order logic via translations. These translations are intended to explain better the exact nature of the modality k says s in BL_S and BL. Finally, it is also an objective of this chapter to introduce the use of authorization logic in modeling access control. The latter, although not a contribution of this chapter or this thesis, will be very helpful to the uninitiated reader in understanding

the rest of this thesis and is best introduced with fewest possible constructs.

At the same time, there are several important aspects of an authorization logic like BL_S that are omitted from this chapter, including proof terms, proof verification, and proof search. These aspects are discussed for the full logic BL in §5 and §6; corresponding aspects for BL_S may be derived as special cases.

History. Two of the fundamental ideas advocated in this chapter and thesis, viz. proof theory in the context of authorization and the emphasis on intuitionistic logic as opposed to classical logic which was the de facto standard in the area for a long time, were first introduced in joint work with Pfenning [67]. The logic used in that paper, called the GP logic here, treats k says $\cdot$ as an indexed lax modality [28, 60, 115]. Although that logic and similar logics by Abadi [5] have been used in several proposals under various names [15, 45, 61, 65, 66, 85, 90, 139], the logic BL used in this thesis contains a weaker modality k says $\cdot$. The switch from the GP logic to BL_S was motivated by three criteria. The first and most important of these is the ability of BL_S (and the inability of the GP logic) to express a specific form of delegation of authority that we call *exclusive delegation.* This form of delegation arises several times in our case study (§8) and is described in §3.1.2. Second, there are simple translations from three existing policy formalisms – the GP logic, Soutei, and a fragment of Binder – into BL_S (§3.5). As a result, BL_S is provably at least as expressive as each of these formalisms (and, in particular, BL_S is at least as expressive as the GP logic). In contrast, we do not know of translations from BL_S, Soutei, or Binder into the GP logic. Third, BL_S admits goal-directed proof search that is complete with respect to its proof rules (§6), which the GP logic may or may not. Goal-directed proof search forms the basis of the automatic proof search tool included in PCFS.

The nature of the says modality in BL is similar to that in the trust management frameworks [118] and the policy language Binder [52], and the name BL is an abbreviation for "Binder" Logic as a tribute to this inspiration. However, there is a significant difference between Binder and BL – the former is a specialized declarative language for writing policies, while the latter is a logic. Most of the technical content in this chapter generalizes previous work on the propositional fragment of BL_S [64, Section 5.5], and on the propositional fragment of a closely related logic DTL_0 [63, 64].

3.1 Syntax and Axioms

BL_S extends first-order intuitionistic logic with a modality k says s, which, as explained earlier, means that principal k states, claims, or supports that formula s is true. Predicates P express relations between terms that are either ground constants a, bound variables x, or applications of uninterpreted function symbols f to ground terms. Terms are classified into sorts σ (sometimes called types). We stipulate at least one sort principal whose elements are represented by the letter k. Formulas s may either be atomic (p, q) or they may be constructed using the usual connectives of predicate logic and the special connective k says s. As a convention, we do not write parenthesis or commas when applying arguments to a predicate, writing an atomic formula as $P\ t_1 \ldots\ t_n$ instead of the more common form

$P(t_1, \ldots, t_n)$ because it makes examples easier to read.

Sorts	σ	::=	$\mathsf{principal} \mid \ldots$
Terms	t, k	::=	$a \mid x \mid f(t_1, \ldots, t_n) \mid \ell$
Predicates	P		
Atoms	p, q	::=	$P\ t_1 \ldots t_n$
Formulas	r, s	::=	$p \mid r \wedge s \mid r \vee s \mid r \supset s \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid \exists x{:}\sigma.s \mid k\ \mathsf{says}\ s$

Negation is not a primitive. If required, it may be defined as $\neg s = (s \supset \bot)$.

Throughout this thesis, the letter Σ denotes a finite partial map from term variables to sorts, concretely represented as $\Sigma = x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n$. We often call Σ a *sorting*. The judgment $\Sigma \vdash t : \sigma$ means that term t has sort σ given the assignment of sorts to variables Σ. We assume the following property of this judgment.

(T-weaken) $\Sigma \vdash t : \sigma$ implies $\Sigma, x{:}\sigma' \vdash t : \sigma$

Further, a stipulated signature specifies the sorts of arguments that function symbols take and the sort that they return, as well as sorts of arguments of predicates, but we do not write the signature explicitly. In a similar manner we elide the details of a formal system of rules to check the well-formedness of syntactic constructs like formulas. Although for most logics well-formedness just means adherence to the grammar, this is not the case for logics considered in this thesis. For instance, well-formedness of $k\ \mathsf{says}\ s$ requires not only that k adhere to the syntax of terms but also that k have sort $\mathsf{principal}$ (in the prevalent sorting). These checks can easily be added to the proof systems presented here, as in prior work [67].

In addition to the judgment $\Sigma \vdash t : \sigma$, proof systems of BL_S are also parameterized by a judgment $\Sigma \vdash k \succeq k'$, read k is stronger than k', or k has more authority in creating policies and credentials than k'. Formally, $\Sigma \vdash k \succeq k'$ has the consequence that $(k\ \mathsf{says}\ s) \supset (k'\ \mathsf{says}\ s)$ for any s that is well-formed in Σ. As a result, the relation $\succeq$ can be used to capture hierarchies in policy administration. It is implicitly assumed that $\Sigma \vdash k : \mathsf{principal}$ and $\Sigma \vdash k' : \mathsf{principal}$ whenever $\Sigma \vdash k \succeq k'$. Although we do not stipulate a definition for the judgment $\Sigma \vdash k \succeq k'$, we require that it satisfy the following properties.

(O-refl) $\Sigma \vdash k \succeq k$

(O-trans) $\Sigma \vdash k \succeq k'$ and $\Sigma \vdash k' \succeq k''$ imply $\Sigma \vdash k \succeq k''$

(O-weaken) $\Sigma \vdash k \succeq k'$ implies $\Sigma, x{:}\sigma \vdash k \succeq k'$

(O-subst) $\Sigma, x{:}\sigma \vdash k \succeq k'$ and $\Sigma \vdash t : \sigma$ imply $\Sigma \vdash k[t/x] \succeq k'[t/x]$

(O-refl) and (O-trans) imply that $\Sigma \vdash k \succeq k'$ defines a preorder on principals. We also assume a distinguished strongest principal ℓ satisfying the following property for every principal k.

(O-loca) $\Sigma \vdash \ell \succeq k$

ℓ is called the "local authority", a term borrowed from the implementation of the language SecPAL [3]. We often abbreviate $\Sigma \vdash k \succeq k'$ to $k \succeq k'$ when Σ is clear from the context or irrelevant. The full logic BL internalizes the order $k \succeq k'$ into the syntax of formulas (§4).

3.1.1 Axiomatic Proof System

Our focus in this chapter, and thesis in general, is on structural proof theory, i.e. natural deduction and sequent calculi. However, for the convenience of readers unfamiliar with these, we describe BL_S's modality k says s using axioms. We write $\Sigma \vdash_{\mathcal{H}} s$ to mean that formula s is valid or provable without hypothesis assuming the sorting Σ for free variables. All variables free in s must occur in the domain of Σ. (The subscript $\mathcal{H}$ represents a Hilbert-style system of proofs). The following axioms and rules for says, together with any complete axiomatization of first-order intuitionistic logic and two additional rules for quantifiers, constitute a deduction system for BL_S. A complete axiomatization for BL_S is presented in Appendix A.

$$\frac{\Sigma \vdash_{\mathcal{H}} s}{\Sigma \vdash_{\mathcal{H}} k \text{ says } s} \quad \text{(N)}$$

$$\Sigma \vdash_{\mathcal{H}} (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2)) \quad \text{(K)}$$
$$\Sigma \vdash_{\mathcal{H}} (k \text{ says } s) \supset k' \text{ says } k \text{ says } s \quad \text{(I)}$$
$$\Sigma \vdash_{\mathcal{H}} k \text{ says } ((k \text{ says } s) \supset s) \quad \text{(C)}$$
$$\Sigma \vdash_{\mathcal{H}} (k \text{ says } s) \supset k' \text{ says } s \text{ if } \Sigma \vdash k \succeq k' \quad \text{(S)}$$

Rule (N) means that each principal states at least all tautologies. Axiom (K) means that the statements of each principal are closed under implication. Together they imply that each $(k \text{ says } \cdot)$ is a *normal* modality (see e.g., [59]). Axiom (I) was first suggested in the context of access control by Abadi [4]. It means that if principal k says s, then every principal k' says that k says s. Axiom (C), an abbreviation for conceit, states that every principal k claims that each of its statements is true. This axiom is peculiar to BL_S and gives its says modality a unique meaning. It is not needed to derive useful consequences from most policies represented in BL_S but is necessary to prove completeness of the axiomatic system with respect to the natural deduction system and the sequent calculus (Theorem 3.13). (S) means that statements of each principal are supported by all weaker principals. In particular, $(\ell \text{ says } s) \supset k \text{ says } s$ for each k and s.

Admissible and inadmissible properties. We list below some theorems and non-theorems in BL_S, primarily to give the reader a better intuition about the nature of the logic. These properties can be established easily using the sequent calculus for BL_S, which we present in §3.2.3. The notation $\rightarrow s$ means that for every formula of the form of s and every Σ whose domain contains the free variables of s, it is the case that $\Sigma \vdash_{\mathcal{H}} s$. $\not\rightarrow s$ denotes its converse. $s \equiv s'$ denotes $(s \supset s') \wedge (s' \supset s)$.

1. $\not\rightarrow \bot$
2. $\not\rightarrow (k \text{ says } s) \supset s$
3. $\not\rightarrow (k \text{ says } \bot) \supset \bot$
4. $\not\rightarrow s \supset k \text{ says } s$

5. $\not\rightarrow (k \text{ says } s) \supset (k \text{ says } k' \text{ says } s)$
6. $\rightarrow (k \text{ says } (s_1 \wedge s_2)) \equiv ((k \text{ says } s_1) \wedge (k \text{ says } s_2))$
7. $\rightarrow ((k \text{ says } s_1) \vee (k \text{ says } s_2)) \supset (k \text{ says } (s_1 \vee s_2))$
8. $\not\rightarrow (k \text{ says } (s_1 \vee s_2)) \supset ((k \text{ says } s_1) \vee (k \text{ says } s_2))$
9. $\not\rightarrow ((k \text{ says } s_1) \supset (k \text{ says } s_2)) \supset (k \text{ says } (s_1 \supset s_2))$
10. $\rightarrow (k \text{ says } \forall x{:}\sigma.s) \supset \forall x{:}\sigma.(k \text{ says } s)$
11. $\not\rightarrow (\forall x{:}\sigma.(k \text{ says } s)) \supset k \text{ says } \forall x{:}\sigma.s$
12. $\rightarrow (\exists x{:}\sigma.(k \text{ says } s)) \supset k \text{ says } \exists x{:}\sigma.s$
13. $\not\rightarrow (k \text{ says } \exists x{:}\sigma.s) \supset \exists x{:}\sigma.(k \text{ says } s)$

(1) is a statement of consistency of BL_S – falsehood is not provable without hypothesis. (2) means that there are statements that principals may make, which are not necessarily true; in particular, $\bot$ is such a statement (3). These two properties are extremely important in an enforcement based on proof-carrying authorization because principals are not constrained in what policy rules they may issue. (4) means that not every true statement is stated by every principal. This may seem counter-intuitive but is necessary to delegate authority in some cases (see §3.5.1 for details). (5) states that even the weaker case of (4) where s has the form k says s' does not hold. As (6) shows, says can be commuted with $\wedge$ without affecting provability. In fact, $\wedge$ is the only connective with which says commutes in this manner. As (7)–(13) show, for every other connective commutation with says preserves provability in exactly one direction.

Example 3.1. We consider a hypothetical example of access control based in the intelligence community. The example is based on a larger case study on the subject, presented in entirety in §8. Suppose that in a hypothetical intelligence agency each file and each individual has a classification level from the ordered set confidential $<$ secret $<$ topsecret. Three distinguished principals participate in access control: admin who has the ultimate authority on granting access, system who is responsible for governing files (e.g., setting their ownership and classification levels), and hr who is responsible for governing individuals (e.g., giving them classification levels). Figure 3.1 shows the policy rules that control access (numbered (1)–(5)) as well as some additional credentials needed to get access in a specific case (numbered (6)–(9)).

In order that principal k may read file f, the following formula must be established from the policies in effect: admin says (`may` k f read). This is possible if k has a classification level above the file (predicate `hasLevelForFile` k f), and k gets permission from the owner of the file. This is captured in policy rule (1) which is created by admin. For readability, we omit all sort annotations from quantifiers. Precisely, policy rule (1) means that (admin says) whenever k has the appropriate clearance level to read file f, system says that k' owns f, and k' says that k may read f, then k may indeed read f. Observe that this rule

Common rules:

admin says $\forall k, k', f.$

(1) $\quad(((\texttt{hasLevelForFile}\ k\ f) \wedge (\mathsf{system\ says}\ (\texttt{owns}\ k'\ f)) \wedge$
$\quad(k'\ \mathsf{says}\ (\texttt{may}\ k\ f\ \mathsf{read}))) \supset \texttt{may}\ k\ f\ \mathsf{read}$

admin says $\forall k, f, l, l'.$

(2) $\quad(((\mathsf{system\ says}\ (\texttt{levelFile}\ f\ l)) \wedge (\mathsf{hr\ says}\ (\texttt{levelPrin}\ k\ l')) \wedge$
$\quad(\texttt{below}\ l\ l')) \supset \texttt{hasLevelForFile}\ k\ f)$

(3) $\ell\ \mathsf{says}\ (\texttt{below}\ \mathsf{confidential\ secret})$

(4) $\ell\ \mathsf{says}\ (\texttt{below}\ \mathsf{secret\ topsecret})$

(5) $\ell\ \mathsf{says}\ (\texttt{below}\ \mathsf{confidential\ topsecret})$

Additional credentials for example scenario:

(6) system says (levelFile secret.txt secret)

(7) system says (owns Alice secret.txt)

(8) hr says (levelPrin Bob topsecret)

(9) Alice says (may Bob secret.txt read)

Figure 3.1: Simplified policies for control of classified information

illustrates how two common policy motifs may be encoded in authorization logic: (a) admin *delegates* control over the predicate owns to principal system, and (b) the file's owner k' is given *discretionary control* over access to it.

Policy rule (2) defines the predicate (hasLevelForFile k f) further in terms of classification levels of k and f (formulas (levelPrin k l) and (levelFile f l), respectively). Observe again that control over levelPrin is delegated to the principal hr whereas control over levelFile is delegated to system. The formula (below l l') captures the order $l < l'$ between classification levels (policy rules (3)–(5)). Since we assume that all principals agree on this order, rules (3)–(5) are stated by the strongest principal, the local authority ℓ.

As an illustration of the use of policy rules (1)–(5), let us assume that file secret.txt owned by Alice is classified at the level secret. Suppose that Bob is an employee cleared at level topsecret, and further that Alice wants to let Bob read file secret.txt. This information is captured in formulas (6)–(9). Using the axioms and rules of BL_S presented earlier (1)–(9) entail admin says (may Bob secret.txt read). Some of the initial steps in this derivation are as follows. First, by instantiating the universal quantifiers in (2), we obtain:

(3.1)
$$\begin{array}{l}\mathsf{admin\ says}\\ \quad(((\mathsf{system\ says}\ (\texttt{levelFile}\ \mathsf{secret.txt\ secret})) \wedge\\ \quad\ (\mathsf{hr\ says}\ (\texttt{levelPrin}\ \mathsf{Bob\ topsecret})) \wedge\\ \quad\ (\texttt{below}\ \mathsf{secret\ topsecret}))\\ \quad\quad \supset \texttt{hasLevelForFile}\ \mathsf{Bob\ secret.txt})\end{array}$$

Basic propositional axioms and (K) yield

(3.2)
$$\begin{array}{l}((\mathsf{admin\ says\ system\ says}\ (\texttt{levelFile}\ \mathsf{secret.txt\ secret})) \wedge\\ \quad(\mathsf{admin\ says\ hr\ says}\ (\texttt{levelPrin}\ \mathsf{Bob\ topsecret})) \wedge\\ \quad(\mathsf{admin\ says}\ (\texttt{below}\ \mathsf{secret\ topsecret})))\\ \quad\quad \supset \mathsf{admin\ says}\ (\texttt{hasLevelForFile}\ \mathsf{Bob\ secret.txt})\end{array}$$

Using axiom (I) with (6) and (8) gives

(3.3) admin says system says (levelFile secret.txt secret)

(3.4) admin says hr says (levelPrin Bob topsecret)

Similarly, axiom (S) on (4) gives

(3.5) admin says (below secret topsecret)

Three applications of modus ponens on (3.2) and (3.3)–(3.5) now yield[1]

(3.6) admin says (hasLevelForFile Bob secret.txt)

The rest of the proof now proceeds similarly: axiom (K) is now applied to (1), then axiom (I) is applied to (7) and (9), and finally admin says (may Bob secret.txt read) is obtained by modus ponens.

It is instructive to observe the role of axiom (I) in injecting the statements (6) and (8) of principals system and hr into the statements of admin (3.3), (3.4). Also, noteworthy is the use of axiom (S) in converting statement (4) made by ℓ to a statement made by admin (3.5). Without axioms (I) and (S) it would be impossible to derive the expected authorization.

Connection to practice. As mentioned in §1.2.2, formulas of the form k says s are special when enforcement is based on proof-carrying authorization because such formulas can be established in two different ways. First, like all other formulas, they may be derived using inference rules and axioms. Second, they can be established *directly* – principal k may write the formula s in a digital certificate and sign it with her private key. In proof-carrying authorization, this digitally signed certificate is evidence that k says s holds. In fact, for a logic like BL$_S$, this is the only primitive way of discharging a hypothesis. From the perspective of enforcement, there is no difference between common policy rules such as (1)–(5) of Figure 3.1, and case specific credentials like (6)–(9). Both are concretely established through digitally signed certificates containing logical formulas.

If proof-carrying authorization is used to enforce these policies, then in order to get access to secret.txt in the above example Bob would give the *proof* which shows that (1)–(9) establish admin says (may Bob secret.txt read) to the reference monitor that is protecting files, together with the certificates that establish (1)–(9). The reference monitor would then check the proof and the certificates, and allow access if both checks succeed. Of course, it may not be very convenient for Bob to either find the proof or represent it using an axiomatic system. For such purposes, structural proof theory may be more appropriate.

3.1.2 Expressible and Inexpressible Policy Idioms

As further illustration of possible use of BL$_S$, and authorization logics in general, we give examples of some idioms that appear often in access policies and discuss whether or not they

[1] Modus ponens is a fundamental rule in axiomatic proof systems which states that $\vdash_{\mathcal{H}} (s \supset s')$ and $\vdash_{\mathcal{H}} s$ imply $\vdash_{\mathcal{H}} s'$.

can be expressed in BL_S. The purpose of this section is explanatory, primarily to show how BL_S can be used in practice. A secondary objective is to compare existing authorization logics and logic-based languages for writing authorization policies (e.g., [4, 8, 23, 26, 49, 52, 67, 88, 118, 143]) in terms of their ability to express these idioms.

Distributed Policies. In distributed systems parts of policies may be created by different individuals. As Example 3.1 illustrates, rules created by different individuals can be represented and combined in BL_S using the says connective. This use of the says connective is not unique to BL_S (although the specific logical behavior of says is). The operator was first introduced by Lampson et al. [8, 88] for exactly this purpose, and has subsequently been adapted in many proposals for expressing distributed authorization policies (e.g. [23, 26, 52, 67, 118]).

Access control lists (ACLs). Although abstracting policies from low level enforcement mechanisms like access control lists is one of the primary reasons to use an authorization logic, ACLs can be encoded in an authorization logic easily. Using notation from Example 3.1, suppose that the principal admin has ultimate authority on deciding access. Then admin may simulate ACLs in the system in any authorization logic including BL_S by issuing one certificate for each entry in the ACLs, e.g.,

$$\begin{array}{c}\mathsf{admin\ says\ (may\ Alice\ foo.txt\ write)}\\ \mathsf{admin\ says\ (may\ Bob\ bar.pdf\ read)}\\ \ldots\end{array}$$

Roles and groups. Roles and groups of principals are used to ease administration of access policies, when a set of individuals have access to exactly the same set of resources. Roles and groups can be expressed very easily in any logic including BL_S. As an example, the case where all members of a group G have read access to all files in the set $f_1, \ldots, f_n$ can be expressed by the following set of n policy rules.

$$\begin{array}{c}\mathsf{admin\ says}\ \forall k.\ ((\mathtt{member}\ k\ G) \supset (\mathtt{may}\ k\ f_1\ \mathsf{read}))\\ \vdots\\ \mathsf{admin\ says}\ \forall k.\ ((\mathtt{member}\ k\ G) \supset (\mathtt{may}\ k\ f_n\ \mathsf{read}))\end{array}$$

where member k G is a predicate which means that k is a member of group G. Members can be added to the group G by signing certificates of the following kind.

$$\begin{array}{c}\mathsf{admin\ says\ (member\ Alice}\ G)\\ \mathsf{admin\ says\ (member\ Bob}\ G)\\ \vdots\end{array}$$

If there are m principals in G and n files, this encoding requires that admin issue $m + n$ certificates, whereas a naive encoding using access control lists would require mn certificates (one for each pair of a file and a principal).

Delegation. In a distributed system a principal k may delegate some or all of its authority in making policies to another principal k'. What this means is that if k' makes a policy rule regarding a subject that has been delegated to it, then k will endorse the rule as well. There are several different kinds of delegation, of which we discuss four here.

The first kind of delegation, which we call *limited delegation*, occurs when a principal k delegates to principal k' authority over a predicate, or in general, over a specific formula s. Such a delegation can be expressed in BL_S as the formula $k \text{ says } ((k' \text{ says } s) \supset s)$. Many examples of this form appear in §8. For this encoding to have its intended effect, i.e. for this formula and $k' \text{ says } s$ to entail $k \text{ says } s$, the logic has to be reasonably strong. In particular, axiom (I) must be admissible in the logic. A number of early authorization logics such as [8, 88] do not satisfy this axiom. Expressing limited delegation in these logics is difficult and requires a rich algebraic structure on principals. The logic $\text{ICL}^{\mathcal{B}}$ enhances the algebraic structure on principals to express limited delegation in a simple manner [65]. A salient observation is that in a limited delegation the delegating principal retains authority over the delegated formula – in the example above, even if k' does not say s, k may directly assert s.

The second kind of delegation, which we call *exclusive delegation*, is a variant of limited delegation, where the delegating principal itself does not have any authority over the formula it delegates. This is illustrated by formulas (1) and (2) in Figure 3.1. In (1), for instance, principal admin delegates authority over the predicate `owns` k' f to the principal system but it is the intent of the policy that admin itself may not make decisions regarding `owns`. Such a delegation can be represented in BL_S using a formula of the form $k \text{ says } ((k' \text{ says } s') \supset s)$. For this encoding to work correctly, the following two properties must hold:

$k \text{ says } ((k' \text{ says } s') \supset s)$ and $k' \text{ says } s'$ should imply $k \text{ says } s$

$k \text{ says } ((k' \text{ says } s') \supset s)$ and $k \text{ says } s'$ should not in general imply $k \text{ says } s$ [2]

Both these properties hold in BL_S. The important observation here is that in the presence of (N) and (K), the first property is equivalent to the axiom (I), whereas the second property imposes a limit on the strength of logic. Interestingly, exclusive delegation is very common in real policies as the case study of access control on classified information demonstrates (§8), but very few authorization logics and logic-based authorization languages can express it. In particular, any logic that admits the axiom $(k \text{ says } s) \supset (k \text{ says } k' \text{ says } s)$ cannot satisfy the second property. (Observe the difference between this axiom and (I)). Many logics including the logic used in the author's prior work, namely the GP logic [67], as well as other logics that treat $k \text{ says } s$ as a lax modality [5, 65] admit the much stronger axiom $s \supset k' \text{ says } s$ and, therefore, cannot express exclusive delegation in a reasonable manner. The logic-based authorization languages Soutei [118] and Binder [52] have says modalities similar to that of BL_S and are capable of expressing exclusive delegation (see §3.5.2 and §3.6). The policy language SecPAL [23] contains a special syntactic construct $k \text{ says } k' \texttt{ cansay } s$ to represent exclusive delegations. As mentioned in the beginning of this chapter, being able to express

[2] In specific cases, this may be inevitable. For example, if $s' \supset s$, then $k \text{ says } s'$ implies $k \text{ says } s$ due to axiom (K).

exclusive delegations is the main reason that this thesis uses the logic BL in place of the GP logic.

In the third type of delegation, called *full delegation* here, complete authority is delegated from one principal to another. If a full delegation is made from k to k', then k' says s should entail k says s for every s. This is a second order property that cannot be expressed in a first-order logic without a specialized construct. The preorder $k \succeq k'$ in BL_S, which may be internalized as a formula in the full logic BL, can be used to express full delegation. Other logics in the past have considered a related formula called "speaks for", often written $k' \Rightarrow k$, with similar effects [8, 65, 88]. In authorization logics with a second order universal quantifier, $k' \Rightarrow k$ can be encoded as $\forall s.\ ((k' \text{ says } s) \supset (k \text{ says } s))$ [5, 65]. Full delegation, if used indiscriminately, can have dangerous consequences in practice. For instance, a principal making a full delegation may not be aware of all predicates that exist in a distributed system and may inadvertently give away authority leading to potential misuse. Further, the presence of full delegation makes automatic proof search much more difficult. Owing to these considerations, in all examples in this thesis that use full delegation, the principal delegated to is the local authority, ℓ (§3.1).

The last form of delegation we consider is *bounded delegation*. In this case, the principal obtaining authority through the delegation has no authority to delegate further. A generalization allows delegation chains of fixed depth. Although bounded delegation is very useful in practice, it cannot be expressed within the logic if the enforcement mechanism is proof-carrying authorization. In particular, bounded delegation cannot be enforced in PCFS. This limitation is the consequence of two fundamental assumptions:

- Principals are unconstrained in policies they create by signing certificates. Consequently, if principal k is delegated control over s, k may always sub-delegate the authority to k' by signing k says $((k' \text{ says } s) \supset s)$.

- *Any* correct proof of authorization is acceptable for access. In the example of the previous point, a proof that used k's certificate would be accepted, provided it were correct otherwise.

Of course, in practice, it is possible to enforce bounded delegation by either restricting what principals may sign, e.g., not allowing k to sign k says $((k' \text{ says } s) \supset s)$ in the first point above, or by restricting the class of acceptable proofs, e.g., disallowing a proof that contains k's delegation certificate. However, both these methods lie outside the proof theory of the logic itself, and are orthogonal to the concerns of this thesis. What would be more aligned to the approach of this thesis is a logic that tracks (counts) the length of a sequence of delegations. The authorization language SecPAL [23] does this to a limited extent.

Dynamic (changing) policies. As mentioned in §1.3, allowed accesses may change in several different ways. Representing and enforcing policies that change over time is quite difficult in logic-based languages, and addressing this challenge is an important goal of both BL and PCFS. Although BL_S is not suitable for representing policies that may change, the full logic BL described in §4 allows representation of authorizations that may expire, as

well those that may depend on system state. §9 goes a step further and discusses a linear extension to BL through which consumable credentials may be represented. Comparisons to other logic-based authorization languages with similar features are provided in the respective chapters.

Ordered rules. A policy idiom commonly found in traditional system configurations is rule precedence. For example, a `.htaccess` file defining the access policy to web pages on an Apache web server may have the following entries.

```
order deny,allow
allow from all
deny from Mallory
```

The first line says that rules denying access take precedence over rules that allow access. The second line allows access to everyone, whereas the third line denies access to the user Mallory. Since the denying rule gets precedence, the net effect of the policy is that all users except Mallory have access. If the precedence were reversed, all users including Mallory would have access. Rule precedence is meaningful only if the policy contains both allow and deny rules.

Although rule precedences similar to the one above are used often in systems, they are extremely difficult to encode in most authorization logics including BL_S and BL. The reason is that in most logics (except those that are non-monotonic), if an authorization follows from a set of policy rules, then it will also follow from any extension of the set. This is a consequence of a fundamental proof-theoretic property called weakening (e.g., Theorem 3.3). On the other hand, policies with rule precedence are incompatible with this property. For instance, in the above policy, the first two rules by themselves would allow access to Mallory, whereas addition of the third rule denies her access.

Consequently, representing policies with rule precedence in a logic requires at least one of the following:

- The logic be non-monotonic, i.e. it not satisfy weakening.

- The informal rules (e.g., from the file above) not map one-to-one to logical formulas.

Both these possibilities are feasible, but are rather antithetic to the idea of distributed authorization. Use of the first possibility requires that the reference monitor enforcing access be aware of all policies ever created; this goes against the basic philosophy of proof-carrying authorization where users present credentials to prove that they have access. In limited cases this may be acceptable, as we do for consumable credentials in §9, but having the reference monitor be aware of all policy rules in a distributed setting may be infeasible. The second solution makes the representation of the policy non-modular – as more denying rules are added, prior allowing rules must be modified to check for absence of conditions. For instance, even though we may encode the rules listed above using a formula of the form $\forall k.\ ((k \neq \texttt{Mallory}) \supset (\texttt{may\_access}\ k))$, if we were to now add a new rule "`deny Baddick`", we would have to *modify* the encoding of the existing rules to the formula $\forall k.\ (((k \neq$

$\texttt{Mallory}) \wedge (k \neq \texttt{Baddick})) \supset (\texttt{may\_access}\ k))$. In order to allow for the possibility of making such changes, the policy must again be centralized. Consequently, we do not attempt to encode rule precedence anywhere in this thesis. Indeed, we are unaware of any logic-based solution that attempts to encode policies with rule precedence. However, there are many formalisms outside logic that allow such policies, e.g., [14, 36, 83, 107].

3.2 Structural Proof Theory

We now turn to the centerpiece of this chapter – structural proof theory for BL_S. By structural proof theory we mean a system of logical inference that admits the so called "structural rules" such as weakening and contraction (Theorems 3.3 and 3.8). More specifically, we are interested in a natural deduction system and a sequent calculus for BL_S (and for the full logic BL in §4). The natural deduction system provides a syntax for proofs that are used directly in enforcement through proof-carrying authorization (§5), while the sequent calculus is useful for many other practical aspects including proof search (§6), and proving several metatheorems later in this chapter. More significantly, we prove several metatheorems about the natural deduction system and the sequent calculus which provide assurance that the logic itself has strong, meaningful foundations. Such an assurance is of great importance in the context of authorization, where a poorly designed logic may easily result in inadvertent consequences and accesses that were not intended by the policy authors. In particular, for BL_S we prove admissibility of cut for the sequent calculus, equivalence of the two inference systems and consistency. Additionally, theorems showing absence of interference among statements of principals can be established easily as in existing work on other logics [5, 67], but this subject is not explored further in this thesis.

Historically, both the natural deduction style of inference and the sequent calculus were first investigated by Gentzen in the context of predicate logic [70]. The specific approach to proof theory followed here is based on Martin-Löf's judgmental method for type theory, where a distinction is made between formulas and judgments [99]. The presentation of the natural deduction system is draws on Pfenning and Davies' work on constructive S4 [115], whereas the presentation of the sequent calculus is inspired by prior joint on multi-modal S4, also done in the context of authorization [66].

3.2.1 Natural Deduction

In Martin-Löf's approach to type theory and logic, formulas are distinguished from judgments. The latter are the objects of knowledge that may be established through proofs. Formulas are the subjects of judgments. For BL_S, we require two basic judgments: $s\ \mathsf{true}$, meaning that formula s is true, and $k\ \mathsf{claims}\ s$, meaning that principal k states or claims that formula s is true. The two basic judgments do not entail each other in general. The judgment $k\ \mathsf{claims}\ s$ is *internalized* by the formula $k\ \mathsf{says}\ s$, which means that $k\ \mathsf{claims}\ s$ is equivalent (at the level of judgments) to $(k\ \mathsf{says}\ s)\ \mathsf{true}$.

Hypothetical judgments and views. Reasoning from hypotheses or assumptions is a basic tenet of logic. For propositional intuitionistic logic the *hypothetical judgment* takes the form $\Gamma \vdash s\ \mathsf{true}$, meaning that the assumptions in Γ entail the judgment $s\ \mathsf{true}$. In the first-order case, a generalization of the form $\Sigma; \Gamma \vdash s\ \mathsf{true}$ is needed. If $\Sigma = x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n$, then $\Sigma; \Gamma \vdash s\ \mathsf{true}$ means that reasoning under the assumptions that each x_i stands for an arbitrary term of sort σ_i, there is a proof which shows that hypotheses Γ entail $s\ \mathsf{true}$.

A distinguishing characteristic of BL_S is that hypothetical reasoning is always performed relative to the claims of a principal k, which we indicate in the hypothetical judgment by writing the latter as $\Sigma; \Gamma \vdash^k s\ \mathsf{true}$. ($s\ \mathsf{true}$ is often abbreviated to s.) Formally, k is called the *view* of the hypothetical judgment, or the view of reasoning. The hypotheses, as usual, are a possibly empty multiset of basic judgments:

Basic judgments	J	$::=$	$s\ \mathsf{true} \mid k\ \mathsf{claims}\ s$	
Sorting	Σ	$::=$	$a_1{:}\sigma_1 \ldots a_n{:}\sigma_n$	
Hypotheses	Γ	$::=$	$J_1 \ldots J_n$	$(n \geq 0)$
Hypothetical judgments			$\Sigma; \Gamma \vdash^k s\ \mathsf{true}$	

Reasoning in BL_S is guided by three basic principles. The first principle, called the *view principle*, describes how the view k affects reasoning.

> **View principle.** While reasoning in view k_0, the assumption $k\ \mathsf{claims}\ s$ entails $s\ \mathsf{true}$ if $k \succeq k_0$.

We incorporate this principle into the natural deduction system by the following rule of inference.

$$\frac{\Sigma \vdash k \succeq k_0}{\Sigma; \Gamma, k\ \mathsf{claims}\ s \vdash^{k_0} s}\mathrm{claims}$$

Based on the view principle, we may define the meaning of the hypothetical judgment $\Sigma; \Gamma \vdash^k s$ precisely as follows:

> *"If $\Sigma = x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n$, then under the assumptions that each x_i stands for an arbitrary term of sort σ_i, and that claims of principals stronger than k are true, there is a proof which shows that Γ logically entail that s is true."*

Although this choice of relativizing hypothetical judgments to claims of principals is non-standard, it seems quite useful from the perspective of access control, where an authorization may succeed or fail, depending on what policies the principal making the decision believes. (A further explanation of the hypothetical judgment in BL_S appears in §3.3.1 after a discussion of metatheory.) Another point to note is that in the above definition of hypothetical judgments, there a *single* proof which is parametric in $x_1, \ldots, x_n$. In particular, case analysis of possible instances of the variables $x_1, \ldots, x_n$ is not an admissible proof rule. This is manifest in Theorem 3.2.

Our second guiding principle, called the *substitution principle*, elaborates the meaning of hypothesis. It states that a hypothesis $s\ \mathsf{true}$ used in a proof may be substituted by an actual proof of the hypothesis. This principle occurs in a similar form in judgmental presentations of other logics also (e.g., [39, 115]).

Substitution principle. $\Sigma; \Gamma \vdash^k s$ and $\Sigma; \Gamma, s \vdash^k s'$ imply $\Sigma; \Gamma \vdash^k s'$

Unlike the view principle which is incorporated directly as a rule in the natural deduction system, the substitution principle, together with the next principle, is established as a theorem (Theorem 3.5).

The third guiding principle, called the *claim principle*, defines the relation between the judgments k claims s and s true. Informally it states that k claims s holds if we can establish s true in the view k from only the claims of other principals. Formally, we define an operator $\Gamma|$ that restricts the hypothesis Γ to the claims of principals.

$$\Gamma| \;=\; \{(k' \text{ claims } s) \in \Gamma\}$$

The claim principle may then be written as follows.

Claim principle. $\Sigma; \Gamma| \vdash^k s$ and $\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s'$ imply $\Gamma \vdash^{k_0} s'$.

The requirement to restrict the hypotheses to substitute k claims s is similar to restrictions that arise for substituting valid hypothesis in constructive S4 [115], and unrestricted assumptions in linear logic [39]. Indeed, k claims s is closely related to the validity judgment from constructive S4. This is described further in §3.4.

Inference rules. The inference rules of the natural deduction system are summarized in Figure 3.2. The most basic inference rule is (hyp). It means that if s true is a hypothesis, then s must be true.

$$\frac{}{\Sigma; \Gamma, s \vdash^k s}\text{hyp}$$

The rule (claims) captures the view principle as described earlier. The remaining rules are directed by the connectives of BL_S, as is the norm in a natural deduction system. For each connective, there are *introduction* rules (marked I) that specify how a proof of the connective may be constructed directly, and *elimination* rules (marked E) that specify how a proof of the connective may be used. In the following we describe briefly the rules for says.

Since (k says s) true internalizes, and hence is equivalent to, the judgment k claims s the claim principle tells us that (k says s) true may be established if we can establish s true in view k using only claims of principals. This is exactly what the rule (saysI) captures:

$$\frac{\Sigma; \Gamma| \vdash^k s}{\Sigma; \Gamma \vdash^{k_0} k \text{ says } s}\text{saysI}$$

The equivalence of (k says s) true and k claims s also implies that we may use (k says s) true by assuming k claims s. This results in the following elimination rule (saysE):

$$\frac{\Sigma; \Gamma \vdash^{k_0} k \text{ says } s \qquad \Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s'}{\Sigma; \Gamma \vdash^{k_0} s'}\text{saysE}$$

$$\frac{}{\Sigma; \Gamma, s \vdash^k s}\mathrm{hyp} \qquad \frac{\Sigma \vdash k \succeq k_0}{\Sigma; \Gamma, k \ \mathsf{claims}\ s \vdash^{k_0} s}\mathrm{claims}$$

$$\frac{\Sigma; \Gamma \vdash^k s \qquad \Sigma; \Gamma \vdash^k s'}{\Sigma; \Gamma \vdash^k s \wedge s'}\wedge\mathrm{I} \qquad \frac{\Sigma; \Gamma \vdash^k s \wedge s'}{\Sigma; \Gamma \vdash^k s}\wedge\mathrm{E}_1 \qquad \frac{\Sigma; \Gamma \vdash^k s \wedge s'}{\Sigma; \Gamma \vdash^k s'}\wedge\mathrm{E}_2$$

$$\frac{\Sigma; \Gamma \vdash^k s}{\Sigma; \Gamma \vdash^k s \vee s'}\vee\mathrm{I}_1 \qquad \frac{\Sigma; \Gamma \vdash^k s'}{\Sigma; \Gamma \vdash^k s \vee s'}\vee\mathrm{I}_2 \qquad \frac{\Sigma; \Gamma \vdash^k s \vee s' \qquad \Sigma; \Gamma, s \vdash^k s'' \qquad \Sigma; \Gamma, s' \vdash^k s''}{\Sigma; \Gamma \vdash^k s''}\vee\mathrm{E}$$

$$\frac{\Sigma; \Gamma, s \vdash^k s'}{\Sigma; \Gamma \vdash^k s \supset s'}\supset\mathrm{I} \qquad \frac{\Sigma; \Gamma \vdash^k s \supset s' \qquad \Sigma; \Gamma \vdash^k s}{\Sigma; \Gamma \vdash^k s'}\supset\mathrm{E}$$

$$\frac{}{\Sigma; \Gamma \vdash^k \top}\top\mathrm{I} \qquad \frac{\Sigma; \Gamma \vdash^k \bot}{\Sigma; \Gamma \vdash^k s}\bot\mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma; \Gamma \vdash^k s}{\Sigma; \Gamma \vdash^k \forall x{:}\sigma.s}\forall\mathrm{I} \qquad \frac{\Sigma; \Gamma \vdash^k \forall x{:}\sigma.s \qquad \Sigma \vdash t : \sigma}{\Sigma; \Gamma \vdash^k s[t/x]}\forall\mathrm{E}$$

$$\frac{\Sigma; \Gamma \vdash^k s[t/x] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Gamma \vdash^k \exists x{:}\sigma.s}\exists\mathrm{I} \qquad \frac{\Sigma; \Gamma \vdash^k \exists x{:}\sigma.s \qquad \Sigma, x{:}\sigma; \Gamma \vdash^k s'}{\Sigma; \Gamma \vdash^k s'}\exists\mathrm{E}$$

$$\frac{\Sigma; \Gamma| \vdash^k s}{\Sigma; \Gamma \vdash^{k_0} k \ \mathsf{says}\ s}\mathsf{saysI} \qquad \frac{\Sigma; \Gamma \vdash^{k_0} k \ \mathsf{says}\ s \qquad \Sigma; \Gamma, k \ \mathsf{claims}\ s \vdash^{k_0} s'}{\Sigma; \Gamma \vdash^{k_0} s'}\mathsf{saysE}$$

Figure 3.2: BL_S: Natural deduction

Rules for the connectives $\wedge$, $\vee$, $\supset$, $\top$, $\bot$, $\forall$, and $\exists$ are standard, with the exception that a view associated with each hypothetical judgment. The view never changes in the rules for any of these connectives and we elide a description of these standard rules. For any syntactic entity Ξ, $\Xi[t/x]$ denotes the standard capture avoiding substitution of term t for the variable x in Ξ. In the rules ($\forall$I) and ($\exists$E), the variable x can occur only in the subderivation above the premise in which it appears in the sorting. We assume that implicit α-renaming may be performed in quantifiers to force this to be the case.

3.2.2 Metatheory of Natural Deduction

Having defined the natural deduction system for BL_S, we now seek to prove general properties of deductions in it. Such properties are called metatheorems. First, we prove the following instantiation theorem, which means that instantiation of parameters (variables in the domain of Σ) preserves provability or, more succinctly, that a proof of $\Sigma; \Gamma \vdash^k s$ is parametric in all variables in the domain of Σ.

Theorem 3.2 (Instantiation)**.** $\Sigma, x{:}\sigma; \Gamma \vdash^k s$ *and* $\Sigma \vdash t : \sigma$ *imply* $\Sigma; \Gamma[t/x] \vdash^{k[t/x]} s[t/x]$

Proof. By induction on the derivation of $\Sigma, x{:}\sigma; \Gamma \vdash^k s$, using properties (T-weaken) and (O-weaken) from §3.1. □

Next, we prove that the structural properties of weakening and contraction are admissible in natural deduction.

Theorem 3.3 (Weakening and contraction)**.** *The following hold.*

1. *(Weakening)* $\Sigma; \Gamma \vdash^k s$ *implies* $\Sigma; \Gamma, J \vdash^k s$.
2. *(Contraction)* $\Sigma; \Gamma, J, J \vdash^k s$ *implies* $\Sigma; \Gamma, J \vdash^k s$.

Further the derivation in the consequent of each statement has a depth no more than that of the antecedent.[3] *(As defined in Section 3.2.1, J denotes an arbitrary judgment of the form s* true *or k* claims *s.)*

Proof. In each case by induction on the given derivation. □

Two important metatheorems already mentioned in §3.2.1 are the substitution principle and the claim principle. Proving these properties needs another important metatheorem called *view subsumption*, which states that weaker views make more formulas provable. Intuitively, view subsumption may be justified directly from the definition of hypothetical judgments.

Theorem 3.4 (View subsumption)**.** $\Sigma \vdash k \succeq k'$ *and* $\Sigma; \Gamma \vdash^k s$ *imply* $\Sigma; \Gamma \vdash^{k'} s$.

Proof. By induction on the derivation of $\Sigma; \Gamma \vdash^k s$, and case analysis of the last rule. The only two interesting cases are shown below.

Case. $\dfrac{\Sigma \vdash k'' \succeq k}{\Sigma; \Gamma, k'' \ \mathsf{claims}\ s \vdash^k s}\mathsf{claims}$

1. $\Sigma \vdash k'' \succeq k$ (Premise)
2. $\Sigma \vdash k \succeq k'$ (Assumption)
3. $\Sigma \vdash k'' \succeq k'$ (1,2; $\succeq$ is a preorder)
4. $\Sigma; \Gamma, k'' \ \mathsf{claims}\ s \vdash^{k'} s$ (Rule (claims) on 3)

Case. $\dfrac{\Sigma; \Gamma| \vdash^{k''} s'}{\Sigma; \Gamma \vdash^k k'' \ \mathsf{says}\ s'}\mathsf{saysI}$

Here $s = k'' \ \mathsf{says}\ s'$.

1. $\Sigma; \Gamma| \vdash^{k''} s'$ (Premise)

[3] The depth of a derivation is defined as the maximum number of inference rules on a path in the derivation that starts from its conclusion and ends at a leaf. Rules needed to establish the auxiliary judgment $\Sigma \vdash t : \sigma$ are not part of BL_S's inference system and do not count towards the depth.

2. $\Sigma; \Gamma \vdash^{k'} k'' \text{ says } s'$ (Rule (saysI) on 1)

□

The following theorem formally states that both the substitution and claim principles hold.

Theorem 3.5 (Substitution and claim)**.** *The following hold.*

1. *(Substitution)* $\Sigma; \Gamma \vdash^{k} s$ *and* $\Sigma; \Gamma, s \vdash^{k} s'$ *imply* $\Sigma; \Gamma \vdash^{k} s'$.
2. *(Claim)* $\Sigma; \Gamma| \vdash^{k} s$ *and* $\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s'$ *imply* $\Sigma; \Gamma \vdash^{k_0} s'$.

Proof. In each case by induction on the second given derivation and case analysis of the last rule in it. The only interesting cases are the following, both in the proof of (2).

Case. $\dfrac{\Sigma \vdash k \succeq k_0}{\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s}$claims

To show: $\Sigma; \Gamma \vdash^{k_0} s$

1. $\Sigma; \Gamma| \vdash^{k} s$ (Assumption)
2. $\Sigma; \Gamma \vdash^{k} s$ (Weakening from Theorem 3.3 on 1)
3. $\Sigma \vdash k \succeq k_0$ (Premise)
4. $\Sigma; \Gamma \vdash^{k_0} s$ (Theorem 3.4 on 2 and 3)

Case. $\dfrac{\Sigma; (\Gamma, k \text{ claims } s)| \vdash^{k'} s'}{\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} k' \text{ says } s'}$saysI

To show: $\Sigma; \Gamma \vdash^{k_0} k' \text{ says } s'$

1. $(\Gamma, k \text{ claims } s)| = \Gamma|, k \text{ claims } s$ (Defn. of $\cdot|$)
2. $\Sigma; \Gamma|, k \text{ claims } s \vdash^{k'} s'$ (Premise and 1)
3. $\Gamma| = (\Gamma|)|$ (Defn. of $\cdot|$)
4. $\Sigma; \Gamma| \vdash^{k} s$ (Assumption)
5. $\Sigma; (\Gamma|)| \vdash^{k} s$ (3,4)
6. $\Sigma; \Gamma| \vdash^{k'} s'$ (i.h. on 5,2)
7. $\Sigma; \Gamma \vdash^{k_0} k' \text{ says } s'$ (Rule (saysI) on 6)

□

3.2.3 Sequent Calculus

Next we develop a sequent calculus for BL_S. As opposed to a natural deduction system where rules modify the conclusion of the hypothetical judgment (the right side of $\vdash^k$), in a sequent calculus rules operate both on the conclusion and the hypotheses. The merit of a sequent calculus lies in the properties of derivations it admits; in particular, the subformula property (Theorem 3.12) is extremely helpful for proving theorems later in this chapter. When compared to a natural deduction system or an axiomatic system, a sequent calculus is also more amenable to automatic proof search, a fact that we exploit for the full logic BL in §6. Finally, the metatheorems of the sequent calculus, in particular the admissibility of cut (Theorem 3.10) provide further evidence of good foundations of the logic.

As for the natural deduction system in §3.2.1, we follow the judgmental method. In fact the structure of hypothetical judgments we use is the same as that for the natural deduction system. We change the entailment symbol from $\vdash^k$ to $\xrightarrow{k}$ to distinguish the two systems where confusion may arise. Hypothetical judgments in the sequent calculus are called sequents.

$$\text{Sequent} \quad ::= \quad \Sigma; \Gamma \xrightarrow{k} s\ \mathsf{true}$$

As before, k is called the view of the sequent and we abbreviate the judgment s true to s when no confusion can arise.[4]

The rules of BL_S's sequent calculus are summarized in Figure 3.3. Two fundamental rules (init) and (claims) relate the different judgments. (init) means that if p true is assumed, then it can be concluded. p must be an atomic formula. It is shown in Theorem 3.11 that a generalization of this rule to arbitrary formulas is admissible in the sequent calculus, i.e. for any formula s, there is a sequent calculus derivation of $\Sigma; \Gamma, s \xrightarrow{k} s$. It should be observed that (init) is the only rule that relates the hypotheses of a sequent to its conclusion. All other rules operate exclusively either on the hypotheses or on the conclusion. (claims) is the sequent calculus equivalent of the rule of same name in the natural deduction system (Figure 3.2). However, in the sequent calculus, the rule works entirely in the hypotheses – the judgment s true derived from k claims s is introduced as an assumption in the premise, rather than a conclusion (as was the case in the natural deduction system).

The remaining rules of the sequent calculus are directed by the connectives of BL_S. For each connective, there are right rules (marked R), which specify how the connective may be decomposed if it appears at the top level in the conclusion of a sequent, and there are left rules (marked L), which specify how the connective may be decomposed in the hypotheses. The right rules are similar to introduction rules of the natural deduction system. The left rules fulfill the purpose of the elimination rules from the natural deduction system, but are not similar to them, since the former decompose connectives in the hypotheses instead of conclusions. However, as Theorem 3.13 shows, the two formulations are equivalent in terms of provability.

[4] Strictly speaking, in a sequent calculus the judgment s true in the hypotheses should be distinguished from that in conclusions. However, this distinction is always evident from the position of the judgment in a sequent, so we use the name true for both.

$$\frac{}{\Sigma;\Gamma, p \xrightarrow{k} p}\text{init} \qquad \frac{\Sigma \vdash k \succeq k_0 \qquad \Sigma;\Gamma, k\ \mathsf{claims}\ s, s \xrightarrow{k_0} r}{\Sigma;\Gamma, k\ \mathsf{claims}\ s \xrightarrow{k_0} r}\text{claims}$$

$$\frac{\Sigma;\Gamma \xrightarrow{k} s \qquad \Sigma;\Gamma \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \wedge s'}\wedge\mathrm{R} \qquad \frac{\Sigma;\Gamma, s \wedge s', s, s' \xrightarrow{k} r}{\Sigma;\Gamma, s \wedge s' \xrightarrow{k} r}\wedge\mathrm{L}$$

$$\frac{\Sigma;\Gamma \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k} s \vee s'}\vee\mathrm{R}_1 \qquad \frac{\Sigma;\Gamma \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \vee s'}\vee\mathrm{R}_2 \qquad \frac{\Sigma;\Gamma, s \vee s', s \xrightarrow{k} r \qquad \Sigma;\Gamma, s \vee s', s' \xrightarrow{k} r}{\Sigma;\Gamma, s \vee s' \xrightarrow{k} r}\vee\mathrm{L}$$

$$\frac{}{\Sigma;\Gamma \xrightarrow{k} \top}\top\mathrm{R} \qquad \frac{}{\Sigma;\Gamma, \bot \xrightarrow{k} r}\bot\mathrm{L}$$

$$\frac{\Sigma;\Gamma, s \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \supset s'}\supset\mathrm{R} \qquad \frac{\Sigma;\Gamma, s \supset s' \xrightarrow{k} s \qquad \Sigma;\Gamma, s \supset s', s' \xrightarrow{k} r}{\Sigma;\Gamma, s \supset s' \xrightarrow{k} r}\supset\mathrm{L}$$

$$\frac{\Sigma, x{:}\sigma;\Gamma \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k} \forall x{:}\sigma.s}\forall\mathrm{R} \qquad \frac{\Sigma;\Gamma, \forall x{:}\sigma.s, s[t/x] \xrightarrow{k} r \qquad \Sigma \vdash t : \sigma}{\Sigma;\Gamma, \forall x{:}\sigma.s \xrightarrow{k} r}\forall\mathrm{L}$$

$$\frac{\Sigma;\Gamma \xrightarrow{k} s[t/x] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Gamma \xrightarrow{k} \exists x{:}\sigma.s}\exists\mathrm{R} \qquad \frac{\Sigma, x{:}\sigma;\Gamma, \exists x{:}\sigma.s, s \xrightarrow{k} r}{\Sigma;\Gamma, \exists x{:}\sigma.s \xrightarrow{k} r}\exists\mathrm{L}$$

$$\frac{\Sigma;\Gamma| \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k_0} k\ \mathsf{says}\ s}\mathsf{saysR} \qquad \frac{\Sigma;\Gamma, k\ \mathsf{says}\ s, k\ \mathsf{claims}\ s \xrightarrow{k_0} r}{\Sigma;\Gamma, k\ \mathsf{says}\ s \xrightarrow{k_0} r}\mathsf{saysL}$$

Figure 3.3: BL_S: Cut-free sequent calculus

The rules for the connectives $\wedge$, $\vee$, $\supset$, $\top$, $\bot$, $\forall$, and $\exists$ are standard, and we do not describe them here. Rule (saysR) is identical to (saysI), except for the difference in the entailment symbol. Rule (saysL) allows decomposition of (k says s) true in the hypotheses by introducing the equivalent equivalent judgment k claims s as an additional assumption. In the rules ($\forall$R) and ($\exists$L), the variable x can occur only in the sub-derivation above the premise in which it appears in the sorting.

Example 3.6. We revisit Example 3.1, showing how the authorization admin says (may Bob secret.txt read) may be derived from the policy rules and credentials (1)–(9) in Figure 3.1. Let the set of formulas (1)–(9) be denoted by Γ. What we seek to show is that $\Sigma;\Gamma \xrightarrow{k}$ admin says (may Bob secret.txt read), where Σ defines the sorts of known constants like Bob, secret.txt, etc. and k is arbitrary. We construct a sequent calculus proof working backwards from this required sequent.

First, using the rule (saysL) from Figure 3.3 several times, we observe that it suffices to

show instead that

$$\Sigma; \Gamma' \xrightarrow{k} \mathsf{admin\ says\ (may\ Bob\ secret.txt\ read)} \tag{3.7}$$

where Γ' is obtained from Γ by replacing all top-level says with claims. Abusing notation slightly, we refer to the formulas in Γ' with the same numbers as the original formulas in Γ. Next, using the rule (saysR) it suffices to show that $\Sigma; \Gamma'| \xrightarrow{\mathsf{admin}} \mathtt{may}\ \mathsf{Bob\ secret.txt\ read}$. Since $\Gamma'| = \Gamma'$, it is enough to prove that

$$\Sigma; \Gamma' \xrightarrow{\mathsf{admin}} \mathtt{may}\ \mathsf{Bob\ secret.txt\ read} \tag{3.8}$$

Using the rule (claims) on hypothesis (1), it suffices to prove

$$\Sigma; \Gamma', r_1 \xrightarrow{\mathsf{admin}} \mathtt{may}\ \mathsf{Bob\ secret.txt\ read} \tag{3.9}$$

where

$$r_1 \quad = \quad \forall k, k', f.\ (((\mathtt{hasLevelForFile}\ k\ f) \wedge (\mathsf{system\ says}\ (\mathtt{owns}\ k'\ f)) \wedge (k'\ \mathsf{says}\ (\mathtt{may}\ k\ f\ \mathsf{read}))) \supset \mathtt{may}\ k\ f\ \mathsf{read})$$

Using rule ($\forall$L) on r_1 thrice to instantiate the universal quantifiers, followed by ($\supset$L) to decompose the implication in r_1 and ($\wedge$R) to decompose the conjunctions, we observe that it suffices to show each of the following.

$$\Sigma; \Gamma', r_1 \xrightarrow{\mathsf{admin}} \mathtt{hasLevelForFile}\ \mathsf{Bob\ secret.txt} \tag{3.10}$$

$$\Sigma; \Gamma', r_1 \xrightarrow{\mathsf{admin}} \mathsf{system\ says}\ (\mathtt{owns}\ \mathsf{Alice\ secret.txt}) \tag{3.11}$$

$$\Sigma; \Gamma', r_1 \xrightarrow{\mathsf{admin}} \mathsf{Alice\ says}\ (\mathtt{may}\ \mathsf{Bob\ secret.txt\ read}) \tag{3.12}$$

The proof of (3.10) follows a pattern similar to the above, except that we must now operate on policy rule (2); the details are omitted here since they provide no new insight. Proofs of (3.11) and (3.12) are similar to each other. As an illustration, we show how (3.12) is established. Using the rule (saysR) and observing that $(\Gamma', r_1)| = \Gamma'$, it suffices to show that

$$\Sigma; \Gamma' \xrightarrow{\mathsf{Alice}} \mathtt{may}\ \mathsf{Bob\ secret.txt\ read} \tag{3.13}$$

Next, we apply the rule (claims) to policy rule (9), reducing the problem to that of proving

$$\Sigma; \Gamma', r_2 \xrightarrow{\mathsf{Alice}} \mathtt{may}\ \mathsf{Bob\ secret.txt\ read} \tag{3.14}$$

where

$$r_2 \quad = \quad \mathtt{may}\ \mathsf{Bob\ secret.txt\ read}$$

Since r_2 equals the conclusion of (3.14), the latter follows immediately from rule (init) and hence this branch closes.

This example illustrates the general pattern of deriving authorizations from policies written in BL_S. The key observation here is that whenever the sequent to be established has a conclusion of the form k says s, e.g., (3.7) and (3.12) above, the (saysR) rule can be used to reduce the problem to that of showing s *in the view* k. The new view k is very important since it allows application of the (claims) rule to promote claims of k to truth, which may not have been possible in earlier views. In the example above, we use the latter step to introduce the assumptions r_1 and r_2 in (3.9) and (3.14) respectively. Truth assumptions like r_1 and r_2 can then be decomposed using left rules. This may result in new goals as, for example, happens above when the implication in r_1 is decomposed. Due to the restriction operator $\Gamma|$, (saysR) also removes truth assumptions from earlier views, e.g., r_1 is removed in (3.13) when the view changes from admin to Alice. This is essential because the principal in the new view may not believe truths from an earlier view – r_1, for example, was stated by admin and Alice may not trust it.

3.2.4 Metatheory of the Sequent Calculus

Some of the metatheorems of the natural deduction system have corresponding analogues in the sequent calculus as well. These include instantiation (Theorem 3.2), weakening and contraction (Theorem 3.3), view subsumption (Theorem 3.4), and instantiation (Theorem 3.2).

Theorem 3.7 (Instantiation)**.** $\Sigma, x{:}\sigma; \Gamma \xrightarrow{k} s$ *and* $\Sigma \vdash t : \sigma$ *imply* $\Sigma; \Gamma[t/x] \xrightarrow{k[t/x]} s[t/x]$

Proof. By induction on the derivation of $\Sigma, x{:}\sigma; \Gamma \xrightarrow{k} s$, using properties (T-weaken) and (O-weaken) from §3.1. □

Theorem 3.8 (Weakening and contraction)**.** *The following hold.*

1. *(Weakening)* $\Sigma; \Gamma \xrightarrow{k} s$ *implies* $\Sigma; \Gamma, J \xrightarrow{k} s$.
2. *(Contraction)* $\Sigma; \Gamma, J, J \xrightarrow{k} s$ *implies* $\Sigma; \Gamma, J \xrightarrow{k} s$.

Further the derivation in the consequent of each statement has a depth no more than that of the antecedent.

Proof. In each case by induction on the given derivation. □

Theorem 3.9 (View subsumption)**.** $\Sigma \vdash k \succeq k'$ *and* $\Sigma; \Gamma \xrightarrow{k} s$ *imply* $\Sigma; \Gamma \xrightarrow{k'} s$.

Proof. By induction on the given derivation of $\Sigma; \Gamma \xrightarrow{k} s$. □

An extremely important theorem in the sequent calculus is admissibility of cut [70], which we state and prove below. Superficially, the statement of the theorem is similar to the substitution and claim principles from natural deduction (Theorem 3.5). However, its implications are different. In particular, admissibility of cut in a sequent calculus can be construed as a proof-theoretic statement of the soundness of the inference system, because together with the identity theorem (Theorem 3.11), it implies that the left rules and right

rules are in harmony with each other. A precise understanding of this intuition can only be obtained by working through the details of the proof of admissibility of cut. Here, it suffices to state that admissibility of cut provides a very strong assurance of good proof-theoretic foundations, which are essential for an authorization logic like BL_S. A second use of the cut theorem is in proving other theorems. For instance it is used to show that axiomatic and natural deduction proofs can be simulated in the sequent calculus (Theorem 3.13).

Theorem 3.10 (Admissibility of cut)**.** *The following hold.*

1. $\Sigma; \Gamma \xrightarrow{k} s$ *and* $\Sigma; \Gamma, s \xrightarrow{k} r$ *imply* $\Sigma; \Gamma \xrightarrow{k} r$
2. $\Sigma; \Gamma| \xrightarrow{k} s$ *and* $\Sigma; \Gamma, k \;\mathsf{claims}\; s \xrightarrow{k_0} r$ *imply* $\Sigma; \Gamma \xrightarrow{k_0} r$

Proof. By simultaneous lexicographic induction, first on the size of the cut formula s, then on the order (2) > (1) on the inductive hypotheses, and then on the depths of the two given derivations. This follows prior work for intuitionistic logic [113] and linear logic [43]. Since the proof in [113] is modular in the connectives, we only need to consider new cases for says and claims. For the benefit of the uninitiated reader we provide some details of the proof, and show some representative cases.

Let the letters $\mathcal{D}$ and $\mathcal{E}$ denote the first and second given derivations in each case. To prove (1) we analyze four exhaustive cases separately: (a) where $\mathcal{E}$ ends in a right rule, (b) where $\mathcal{E}$ ends in a left rule but the cut is not principal (i.e. s is not the subject of the last rule in $\mathcal{E}$), (c) where $\mathcal{D}$ ends in a left rule, and (d) where $\mathcal{E}$ ends in a left rule, $\mathcal{D}$ ends in a right rule and the cut is principal (i.e. s is the subject of the last rule of $\mathcal{E}$. Of these (a), (b), and (c) are straightforward. We show here one new case in (d), namely the principal cut of the says connective.

Case. $\mathcal{D} = \dfrac{\Sigma; \Gamma| \xrightarrow{k} s}{\Sigma; \Gamma \xrightarrow{k_0} k \;\mathsf{says}\; s}\mathsf{saysR} \qquad \mathcal{E} = \dfrac{\Sigma; \Gamma, k \;\mathsf{says}\; s, k \;\mathsf{claims}\; s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \;\mathsf{says}\; s \xrightarrow{k_0} r}\mathsf{saysL}$

and the cut judgment is k says s. To show: $\Sigma; \Gamma \xrightarrow{k_0} r$.

1. $\Sigma; \Gamma, k \;\mathsf{claims}\; s \xrightarrow{k_0} r$ (i.h.(1) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Gamma \xrightarrow{k_0} r$ (i.h.(2) of premise of $\mathcal{D}$ and 1)

The application of i.h. in the first step is justified because $\mathcal{E}$ gets smaller. Even though the derivation obtained from step 1 is potentially larger than $\mathcal{E}$, the use of i.h. in the second step is justified because the cut formula s is smaller, and the induction is lexicographic first in the size of this formula, and second in the size of the derivations.

To prove (2) we case analyze the last rule in the derivation of $\mathcal{E}$, distinguishing principal and non-principal cuts when the last rule is a left rule. The only two interesting cases are shown below.

Case. $\mathcal{E} = \dfrac{\Sigma \vdash k \succeq k_0 \qquad \Sigma; \Gamma, k \;\mathsf{claims}\; s, s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \;\mathsf{claims}\; s \xrightarrow{k_0} r}\mathsf{claims}$

and the cut judgment is k claims s. To show: $\Sigma; \Gamma \xrightarrow{k_0} r$.

1. $\Sigma; \Gamma, s \xrightarrow{k_0} r$ (i.h.(2) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Gamma \xrightarrow{k} s$ (Weakening from Theorem 3.8 on $\mathcal{D}$)
3. $\Sigma; \Gamma \xrightarrow{k_0} s$ (Theorem 3.9 on 2)
4. $\Sigma; \Gamma \xrightarrow{k_0} r$ (i.h.(1) on 3,1)

The use of the i.h. in the last step is justified because of the assumed order (2) > (1) among the inductive hypotheses.

Case. $\mathcal{E} = \dfrac{\Sigma; (\Gamma, k \text{ claims } s)| \xrightarrow{k'} s'}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{k_0} k' \text{ says } s'}\text{saysR}$

and the cut judgment is k claims s. To show: $\Sigma; \Gamma \xrightarrow{k_0} k'$ says s'.

1. $(\Gamma, k \text{ claims } s)| = \Gamma|, k \text{ claims } s$ (Defn. of $\cdot|$)
2. $\Sigma; \Gamma|, k \text{ claims } s \xrightarrow{k'} s'$ (Premise of $\mathcal{E}$ and 1)
3. $(\Gamma|)| = \Gamma|$ (Defn. of $\cdot|$)
4. $\Sigma; (\Gamma|)| \xrightarrow{k} s$ ($\mathcal{D}$ and 3)
5. $\Sigma; \Gamma| \xrightarrow{k'} s'$ (i.h.(2) on 4,2)
6. $\Sigma; \Gamma \xrightarrow{k_0} k'$ says s' (Rule (saysR) on 5)

□

Another meta-property of the sequent calculus is the following identity theorem, which generalizes the (init) rule from atomic formulas p to arbitrary formulas s. Like admissibility of cut, the identity theorem also provides confidence that the left and right rules of the sequent calculus fit well with each other.

Theorem 3.11 (Identity)**.** $\Sigma; \Gamma, s \xrightarrow{k} s$ *for each* s.

Proof. By induction on s. □

The last metatheorem about the sequent calculus that we prove here is the subformula property. This property states that if we look at any sequent calculus proof of $\Sigma; \Gamma \xrightarrow{k} s$, then the only formulas arising in this proof are subformulas of formulas already present in Γ, s. Intuitively, this property holds because every sequent calculus rule when read from the conclusion to premises only decomposes formulas. Hence, proceeding backwards, the formulas always get smaller. Although we will not have occasion to use this theorem directly, the idea of having only subformulas is used implicitly in the proofs of several other theorems, including those involving translations between logics later in this chapter (§3.5).

Formally, we define the subformula relation $s \sqsubseteq s'$ as the least relation that is reflexive, transitive, closed under applications of all logical connectives (congruent), and includes the following relations.

$$s \sqsubseteq s \wedge s' \qquad s' \sqsubseteq s \wedge s' \qquad s \sqsubseteq s \vee s' \qquad s' \sqsubseteq s \vee s' \qquad s \sqsubseteq s \supset s' \qquad s' \sqsubseteq s \supset s'$$

$$s[t/x] \sqsubseteq \forall x{:}\sigma.s \qquad s[t/x] \sqsubseteq \exists x{:}\sigma.s \qquad s \sqsubseteq k \text{ says } s$$

We further extend the relation to judgments by requiring that k claims $s \sqsubseteq k$ says s and $s \sqsubseteq k$ claims s, and taking the reflexive, transitive, and congruence closure again.

Theorem 3.12 (Subformula property)**.** *Suppose the sequent $\Sigma'; \Gamma' \xrightarrow{k'} s'$ appears in a proof of the sequent $\Sigma; \Gamma \xrightarrow{k} s$. Then for each judgment J' in Γ', s', there is a judgment J in Γ, s such that $J' \sqsubseteq J$.*

Proof. By induction on the derivation of $\Sigma; \Gamma \xrightarrow{k} s$. □

3.3 Equivalence of Proof Systems

We have presented three different proof systems for BL_S: an axiomatic system in §3.1.1, a natural deduction system in §3.2.1, and a sequent calculus in §3.2.3. Now we show that, despite their vast differences, they establish the same judgments. Formally, we show that proofs in each system can be simulated in the other two. In order to represent hypothetical judgments in the axiomatic system, we define a mapping $\overline{\cdot}$ from judgments and hypotheses to formulas.

$$\begin{array}{rcl} \overline{s \text{ true}} & = & s \\ \overline{k \text{ claims } s} & = & k \text{ says } s \\ \overline{J_1, \ldots, J_n} & = & \overline{J_1} \wedge \ldots \wedge \overline{J_n} \end{array}$$

Theorem 3.13 (Equivalence)**.** *The following are equivalent for any Σ, Γ, k and s.*

1. *$\Sigma; \Gamma \xrightarrow{k} s$ in the sequent calculus.*
2. *$\Sigma; \Gamma \vdash^k s$ in the natural deduction system.*
3. *$\Sigma \vdash_{\mathcal{H}} k$ says $(\overline{\Gamma} \supset s)$ in the axiomatic system.*

Proof. We show that $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$.

Proof of 1 ⇒ 2. To show that every sequent calculus proof can be simulated in the natural deduction system we induct on proofs of $\Sigma; \Gamma \xrightarrow{k} s$ and case analyze the last rule in the derivation. This is fairly standard, and we show here only the new cases involving claims and says.

Case. $\dfrac{\Sigma \vdash k \succeq k_0 \qquad \Sigma; \Gamma, k \text{ claims } s, s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{k_0} r}\text{claims}$

To show: $\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} r$

1. $\Sigma; \Gamma, k \ \mathsf{claims}\ s, s \vdash^{k_0} r$ (i.h. on premise)
2. $\Sigma \vdash k \succeq k_0$ (Premise)
3. $\Sigma; \Gamma, k \ \mathsf{claims}\ s \vdash^{k_0} s$ (Rule (claims) on 2)
4. $\Sigma; \Gamma, k \ \mathsf{claims}\ s \vdash^{k_0} r$ (Theorem 3.5 on 3,1)

Case. $\dfrac{\Sigma; \Gamma| \xrightarrow{k} s}{\Sigma; \Gamma \xrightarrow{k_0} k \ \mathsf{says}\ s}\mathsf{saysR}$

To show: $\Sigma; \Gamma \vdash^{k_0} k \ \mathsf{says}\ s$

1. $\Sigma; \Gamma| \vdash^{k} s$ (i.h. on premise)
2. $\Sigma; \Gamma \vdash^{k_0} k \ \mathsf{says}\ s$ (Rule (saysI) on 1)

Case. $\dfrac{\Sigma; \Gamma, k \ \mathsf{says}\ s, k \ \mathsf{claims}\ s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \ \mathsf{says}\ s \xrightarrow{k_0} r}\mathsf{saysL}$

To show: $\Sigma; \Gamma, k \ \mathsf{says}\ s \vdash^{k_0} r$

1. $\Sigma; \Gamma, k \ \mathsf{says}\ s, k \ \mathsf{claims}\ s \vdash^{k_0} r$ (i.h. on premise)
2. $\Sigma; \Gamma, k \ \mathsf{says}\ s \vdash^{k_0} k \ \mathsf{says}\ s$ (Rule (hyp))
3. $\Sigma; \Gamma, k \ \mathsf{says}\ s \vdash^{k_0} r$ (Rule (saysE) on 2,1)

Proof of 2 $\Rightarrow$ 3. Proving that the natural deduction system can be simulated in the axiomatic system requires many lemmas about the latter. These details and the proof are covered in Appendix A. In particular, see Lemma A.2.

Proof of 3 $\Rightarrow$ 1. First we prove that if $\Sigma \vdash_{\mathcal{H}} s$, then for every k, $\Sigma; \cdot \xrightarrow{k} s$ by showing that every axiom and rule in the axiomatic system can be simulated in the sequent calculus. This is straightforward but requires admissibility of cut (Theorem 3.10) as well as identity (Theorem 3.11); details are in Appendix A (Lemma A.3).

Next we complete the proof assuming this fact. Suppose that $\Sigma \vdash_{\mathcal{H}} k \ \mathsf{says}\ (\overline{\Gamma} \supset s)$. It follows that $\Sigma; \cdot \xrightarrow{k} k \ \mathsf{says}\ (\overline{\Gamma} \supset s)$. The only rule that can be used to derive this sequent is (saysR). Hence, the premise of the rule, i.e $\Sigma; \cdot \xrightarrow{k} \overline{\Gamma} \supset s$ must also hold. Again, the only rule that can be used to derive this sequent is ($\supset$R). From the premise of this rule we get $\Sigma; \overline{\Gamma} \xrightarrow{k} s$. Now observe that $\Sigma; \Gamma \xrightarrow{k} \overline{\Gamma}$. Using Theorem 3.10 on the last two sequents we get $\Sigma; \Gamma \xrightarrow{k} s$ as required. $\square$

3.3.1 On the Nature of Hypothetical Judgments in BL_S

According to Theorem 3.13, the hypothetical judgment $\Sigma; \Gamma \vdash^k s$ (and the sequent $\Sigma; \Gamma \xrightarrow{k} s$) is equivalent to the formula k says $(\overline{\Gamma} \supset s)$ in the axiomatic system. This has two obvious consequences. First, the natural deduction system and the sequent calculus correspond only to a fragment of the axiomatic system, namely the one consisting of formulas that begin with k says $\cdot$. Second, and perhaps more significantly, the view k of a hypothetical judgment applies to the entire hypothetical judgment, not just the conclusion.

The latter means that hypothetical reasoning is always relativized to the claims of the principal in the view. The truth of a formula is also relative to the view; whenever the view changes in a natural deduction or sequent calculus proof (in the rules (saysI) and (saysR)), judgments of the form s true in the hypotheses are erased by the operator $\Gamma|$. These ideas were illustrated in Example 3.6. Whereas this relativization of truth may seem unusual for a logic, it is quite useful in the context of authorization. Indeed, in practice, an authorization will succeed or fail based solely on what the authorizer can be convinced of. This is precisely the case in BL_S.

There is also an extension of the structural proof theory of BL_S that contains a "pure" hypothetical judgment $\Sigma; \Gamma \vdash s$. All rules of natural deduction, except (claims), are allowed for this hypothetical judgment. Further, the new hypothetical judgment in the extension corresponds exactly to the axiomatic system. Since this extension is not useful from the perspective of authorization we do not pursue it in detail here.

3.4 Relation to the Modal Logic Constructive S4

Note: This section assumes basic familiarity with modal logic, and is likely to be relevant only to readers familiar with it. Uninterested readers may skip this section without affecting readability of the rest of this thesis.

Since k says s is a modality, a natural question to ask is whether BL_S is related to other existing modal logics. The answer to this question is affirmative: BL_S is very closely connected to the modal logic constructive S4 (CS4) [11, 29, 115]. The latter is an intuitionistic version of the modal logic S4 whose semantics and proof theory have been explored extensively in the past. In this section, we establish connections between the propositional fragment of BL_S (i.e. the fragment without any quantifiers) and the fragment of CS4 without the possibility modality. The restriction to the propositional case is motivated by the fact that propositional CS4 has a well-studied proof theory. Precisely, we show the following.

- Propositional BL_S, when restricted to only one principal (say ℓ) reduces to CS4 without the possibility modality $\Diamond$. In particular, in this case the modality ℓ says $\cdot$ behaves exactly like $\Box$ in CS4.

- The translation from propositional BL_S to CS4 that maps k says s to $\Box(g_k \supset s)$ (where, for each k, g_k is a distinguished atomic formula in CS4) and all other connectives to themselves is an embedding, i.e. it is sound and complete in terms of

provability. This shows that k says s in BL_S is similar in nature to a necessitation modality.

Constructive S4. CS4 is a propositional intuitionistic (constructive) modal logic with the usual modalities of necessitation $\Box$ and possibility $\Diamond$. Here, we are concerned with CS4 without $\Diamond$. A Hilbert style proof system for this logic consists of any axiomatization of intuitionistic propositional logic, and the following rules and axioms for $\Box s$ [11].

$$\frac{\vdash s}{\vdash \Box s} \qquad \text{(nec)}$$

$$\begin{array}{ll} \vdash (\Box(s \supset s')) \supset ((\Box s) \supset (\Box s')) & \text{(K)} \\ \vdash (\Box s) \supset \Box\Box s & \text{(4)} \\ \vdash (\Box s) \supset s & \text{(T)} \end{array}$$

A natural deduction system for CS4 was described by Pfenning and Davies [115], and a sequent calculus for a generalization with indexed modalities and linearity appeared in prior work [66]. Further, Alechina et al. have studied Kripke and categorical semantics of CS4 [11].

BL_S as a generalization of CS4. An obvious translation from CS4 to BL_S is to map $\Box s$ to ℓ says s and all other connectives to themselves. Remarkably, this simple translation is both sound and complete. Another way to look at this translation is to say that in the degenerate case where there is only one principal (say ℓ) in BL_S, the sole modality ℓ says s behaves exactly like the necessitation modality $\Box s$ from CS4. In fact, in this degenerate case the natural deduction system for BL_S (Figure 3.2) reduces to the judgmental natural deduction system for CS4 developed by Pfenning and Davies [115]. Similarly, the sequent calculus (Figure 3.3) reduces to a corresponding calculus for CS4 (e.g., [66]).

Formally, let $\ulcorner \cdot \urcorner$ be the translation from CS4 formulas to propositional BL_S formulas that maps $\Box s$ to ℓ says $\ulcorner s \urcorner$ and all other connectives to themselves. Then the following theorem shows that BL_S generalizes propositional CS4.

Theorem 3.14. *In the special case where there is only one principal ℓ in BL_S, the following are equivalent for any CS4 formula s.*

1. $\vdash s$ *in CS4.*
2. $\cdot;\cdot \vdash^{\ell} \ulcorner s \urcorner$ *in the natural deduction system of Figure 3.2.*

Proof. This result follows from the observation that if there is only principal ℓ, then the natural deduction system of BL_S in Figure 3.2 reduces to the natural deduction system of CS4 [115]. This is because with only one principal, views are irrelevant and the (claims) rule can be applied at all times. Hence, the judgment ℓ claims s corresponds to the judgment called s valid in [115]. Due to this reduction of the natural deduction system, $\Box$ in CS4 behaves exactly like ℓ says $\cdot$ in this restricted BL_S. $\Box$

Translation from propositional BL_S to CS4. Next we consider a translation from propositional BL_S to CS4. For each principal k in propositional BL_S, let g_k be a distinguished atomic formula in CS4 that does not appear in BL_S. The following translation $\ulcorner\cdot\urcorner$ maps k says s to $\Box(g_k \supset \ulcorner s \urcorner)$ and all other connectives to themselves.

$$\begin{array}{rcl} \ulcorner p \urcorner & = & p \\ \ulcorner s \wedge s' \urcorner & = & \ulcorner s \urcorner \wedge \ulcorner s' \urcorner \\ \ulcorner s \vee s' \urcorner & = & \ulcorner s \urcorner \vee \ulcorner s' \urcorner \\ \ulcorner s \supset s' \urcorner & = & \ulcorner s \urcorner \supset \ulcorner s' \urcorner \\ \ulcorner \top \urcorner & = & \top \\ \ulcorner \bot \urcorner & = & \bot \\ \ulcorner k \text{ says } s \urcorner & = & \Box(g_k \supset \ulcorner s \urcorner) \end{array}$$

The important part of the translation is the mapping of k says s to $\Box(g_k \supset \ulcorner s \urcorner)$. The formula g_k on the left of the implication acts as a "guard" on $\ulcorner s \urcorner$, and recovers the effect of the view associated with hypothetical judgments in BL_S: $\ulcorner s \urcorner$ can be obtained from $g_k \supset \ulcorner s \urcorner$ only if g_k is true. By design, our translation ensures that g_k is true if and only if we are reasoning in a view weaker than k.

Define the set of formulas $\mathcal{O} = \{\Box(g_k \supset g_{k'}) \mid k' \succeq k\}$.[5] $\mathcal{O}$ captures the preorder $\succeq$ between principals as implications between the representations of principals as atomic formulas. The following theorem states the correctness property for the translation. (We abuse notation slightly and use $\mathcal{O}$ to also represent the formula obtained by taking the conjunction of all formulas in the set $\mathcal{O}$.)

Theorem 3.15 (Correctness)**.** $\cdot;\cdot \vdash^k s$ *in* BL_S *if and only if* $\vdash \mathcal{O} \supset (g_k \supset \ulcorner s \urcorner)$ *in CS4.*

Proof. Soundness ("only if" direction) follows by an induction on proofs in BL_S. We must generalize the induction hypothesis to state that $\cdot;\Gamma \vdash^k s$ implies $\vdash \mathcal{O} \supset ((\overline{\ulcorner \Gamma \urcorner} \wedge g_k) \supset \ulcorner s \urcorner)$. Completeness ("if" direction) follows by showing that CS4 sequent calculus proofs of translated formulas can be simulated in BL_S. See [64, Theorem 5.9] for details. □

3.5 Translations from the GP Logic and Soutei to BL_S

In this section we explore formal connections between BL_S and two existing logic-based authorization formalisms, namely the GP logic [67] and Soutei [118]. We prove that both frameworks can be embedded in BL_S in a sound and complete manner. Hence, any authorization policy expressible in either the GP logic or Soutei is also expressible in BL_S. Proofs of correctness of both translations make extensive use of the sequent calculus theory of BL_S. In particular, they rely on the subformula property (Theorem 3.12).

3.5.1 Translation from the GP Logic

The GP logic was first described in joint work with Pfenning [67]; a second-order variant was described independently by Abadi under the name CDD [5]. The syntax of formulas in

[5] Since we are considering only the propositional fragment of BL_S, all principals are ground terms and, hence, the sorting Σ in the judgment $\Sigma \vdash k \succeq k'$ is irrelevant. So we abbreviate the latter to $k \succeq k'$.

the GP logic is the same as that in BL_S, but the says modality is interpreted as a family of lax modalities [28, 60] indexed by principals. The logic is quite expressive, has a simple and well studied proof theory, and together with CDD forms the basis of a lot of subsequent research [61, 65, 66, 85, 90, 139]. Interestingly, it can be embedded into BL_S simply by prefixing all connectives with ℓ says $\cdot$. This translation is inspired by Gödel's translation from intuitionistic logic to modal S4, where a $\Box$ is put before each connective [74]. In the following we describe GP logic briefly and show that this translation to BL_S is a logical embedding.

GP Logic. Formulas of GP logic have the same syntax as formulas of BL_S. For simplicity, we consider here the fragment that was presented in prior work [67]. This fragment contains only atomic formulas, $\supset$, $\bot$, $\forall$, and says. The translation presented here extends to other connectives easily. We use the letters A, B, C to denote formulas in GP logic.

$$A, B, C \quad ::= \quad p \mid A \supset B \mid \bot \mid \forall x{:}\sigma.A \mid k \text{ says } A$$

The modality k says A in GP logic is treated as a monad. It is defined by the following axioms.

$$\begin{array}{ll} \vdash A \supset (k \text{ says } A) & \text{(unit)} \\ \vdash (k \text{ says } (A \supset B)) \supset ((k \text{ says } A) \supset k \text{ says } B) & \text{(K)} \\ \vdash (k \text{ says } k \text{ says } A) \supset k \text{ says } A & \text{(C4)} \end{array}$$

Unlike BL_S, there is no order between principals in GP logic. A sequent calculus for the logic from prior work [67] is reproduced in Figure 3.4. The basic judgments used are A true (abbreviated to A) and k affirms A. The latter is internalized by the connective k says A. Sequents have the form $\Sigma; \Gamma \longrightarrow \gamma$. There are no views. The hypotheses Γ are a multiset of assumptions of the form A true. Conclusions γ may be of either of the two forms A true and k affirms A. For details of the sequent calculus we refer the reader to prior work. The important points to observe are that all left rules except (saysL) apply to all possible conclusions, whereas in the rule (saysL) the conclusion must have the form k affirms B, and this k must match the k in the principal formula k says A on the left.

Translation from the GP logic to BL_S. Let pr be the sort of principals in GP logic. Assume that ℓ is not in pr, and that the sort of principals in BL_S, principal, contains all principals in pr and ℓ (but no others). Thus pr is a subsort of principal. Further assume that unequal principals in pr are not related to each other in the preorder $\succeq$ in BL_S, and that the sort principal does not appear in formulas of GP logic.

The translation $\ulcorner A \urcorner$ from formulas and sequents of GP logic to those of BL_S is defined in Figure 3.5. On formulas the translation adds the prefix ℓ says $\cdot$ before all connectives of the except for the top most connective. On sequents the translation adds the prefix ℓ claims $\cdot$ to all hypotheses, and sets the view and conclusion depending on the conclusion of the GP logic sequent.

$$\frac{}{\Sigma; \Gamma, p \to p}\text{init} \qquad \frac{\Sigma; \Gamma \to A}{\Sigma; \Gamma \to k \text{ affirms } A}\text{affirms}$$

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

$$\frac{}{\Sigma; \Gamma, \bot \to \gamma}\bot\text{L}$$

$$\frac{\Sigma, x{:}\sigma; \Gamma \to A}{\Sigma; \Gamma \to \forall x{:}\sigma.A}\forall\text{R} \qquad \frac{\Sigma; \Gamma, \forall x{:}\sigma.A, A[t/x] \to \gamma}{\Sigma; \Gamma, \forall x{:}\sigma.A \to \gamma}\forall\text{L}$$

$$\frac{\Sigma; \Gamma \to k \text{ affirms } A}{\Sigma; \Gamma \to k \text{ says } A}\text{saysR} \qquad \frac{\Sigma; \Gamma, k \text{ says } A, A \to k \text{ affirms } B}{\Sigma; \Gamma, k \text{ says } A \to k \text{ affirms } B}\text{saysL}$$

Figure 3.4: Sequent calculus for GP logic (reproduced from [67])

Correctness of the translation. The translation $\ulcorner \cdot \urcorner$ is sound and complete, i.e. a sequent is provable in GP logic if and only if its translation is provable in BL_S. Soundness, the "only if" direction is easy to establish by induction on sequent calculus proofs in GP logic.

Theorem 3.16 (Soundness)**.** *If* $\Sigma; \Gamma \to \gamma$ *in GP logic, then* $\ulcorner \Sigma; \Gamma \to \gamma \urcorner$ *in* BL_S*.*

Proof. By induction on the given GP logic proof of $\Sigma; \Gamma \to \gamma$. See Appendix A for details (Theorem A.4). □

The converse of this theorem, completeness, is harder to prove. First, we define a translation $|s|$ from a fragment of BL_S larger than the image of $\ulcorner \cdot \urcorner$ back to GP logic, such that $|\ulcorner \cdot \urcorner|$ is the identity translation. Then, we prove that whenever a sequent is provable in BL_S, its translation under $|\cdot|$ is provable in GP logic. This immediately implies completeness. The subformula property of the sequent calculus (Theorem 3.12) is implicitly used in this proof. It ensures that proofs of sequents in the domain of $|\cdot|$ only contain sequents that are also in its domain, so that we can always apply the induction hypothesis. Although this style of proving completeness of translations has been used in the past, its details are specific to each case, so we discuss them here. The inverse translation $|\cdot|$ is defined in Figure 3.6.

Lemma 3.17 (Composition)**.** *The following hold.*

1. $|\ulcorner A \urcorner| = A$
2. $|\ulcorner \Sigma; \Gamma \to \gamma \urcorner| = \Sigma; \Gamma \to \gamma$

Formulas A

$$
\begin{array}{rcl}
\ulcorner p \urcorner & = & p \\
\ulcorner A \supset B \urcorner & = & (\ell \ \mathsf{says} \ \ulcorner A \urcorner) \supset (\ell \ \mathsf{says} \ \ulcorner B \urcorner) \\
\ulcorner \bot \urcorner & = & \bot \\
\ulcorner \forall x{:}\sigma.A \urcorner & = & \forall x{:}\sigma.\ \ell \ \mathsf{says} \ \ulcorner A \urcorner \\
\ulcorner k \ \mathsf{says} \ A \urcorner & = & k \ \mathsf{says} \ \ell \ \mathsf{says} \ \ulcorner A \urcorner
\end{array}
$$

Hypotheses Γ

$$
\ulcorner A_1 \ \mathsf{true}, \ldots, A_n \ \mathsf{true} \urcorner \ = \ \ell \ \mathsf{claims} \ \ulcorner A_1 \urcorner, \ldots, \ell \ \mathsf{claims} \ \ulcorner A_n \urcorner
$$

Sequents

$$
\begin{array}{rcl}
\ulcorner \Sigma; \Gamma \to B \ \mathsf{true} \urcorner & = & \Sigma; \ulcorner \Gamma \urcorner \xrightarrow{\ell} \ulcorner B \urcorner \ \mathsf{true} \\
\ulcorner \Sigma; \Gamma \to k \ \mathsf{affirms} \ B \urcorner & = & \Sigma; \ulcorner \Gamma \urcorner \xrightarrow{k} \ell \ \mathsf{says} \ \ulcorner B \urcorner \ \mathsf{true}
\end{array}
$$

Figure 3.5: Translation from GP logic to BL_S

Proof. By induction on the syntax of GP logic. □

Lemma 3.18 (Simulation)**.** *Suppose* $\Sigma \vdash k \succeq k'$ *implies* $k = k'$ *for every* k *and* k' *in the sort* pr*. Then, whenever* $\Sigma; \Gamma \xrightarrow{k} s \ \mathsf{true}$ *is in the domain of* $|\cdot|$ *and provable in* BL_S*, it is the case that* $|\Sigma; \Gamma \xrightarrow{k} s \ \mathsf{true}|$ *is provable in GP logic.*

Proof. By induction on the given BL_S derivation of $\Sigma; \Gamma \xrightarrow{k} s \ \mathsf{true}$ and case analysis of its last rule. Since the translation $|\cdot|$ on sequents is defined based on whether the view is ℓ or not, we further distinguish cases based on the view. Some representative cases are shown here. We frequently use the structural properties of weakening and contraction for the sequent calculus of GP logic. These can be proved easily by induction on sequent calculus derivations.

Case. $\dfrac{}{\Sigma; \Gamma, p \xrightarrow{k} p}\,\text{init}$ $(k \neq \ell)$

To show: $\Sigma; |\Gamma|, p \to k \ \mathsf{affirms} \ p$

1. $\Sigma; |\Gamma|, p \to p$ (Rule (init))
2. $\Sigma; |\Gamma|, p \to k \ \mathsf{affirms} \ p$ (Rule (affirms) on 1)

Case. $\dfrac{}{\Sigma; \Gamma, p \xrightarrow{\ell} p}\,\text{init}$

Formulas s

$$
\begin{array}{rcll}
|p| & = & p & \\
|s_1 \supset s_2| & = & |s_1| \supset |s_2| & \\
|\bot| & = & \bot & \\
|\forall x{:}\sigma.s| & = & \forall x{:}\sigma.|s| & \sigma \neq \mathsf{principal} \\
|k \text{ says } s| & = & \begin{cases} k \text{ says } |s| & k \neq \ell \\ |s| & k = \ell \end{cases} &
\end{array}
$$

Hypotheses Γ

$$
\begin{array}{rcll}
|s \text{ true }| & = & |s| \text{ true} & \\
|k \text{ claims } s| & = & \begin{cases} (k \text{ says } |s|) \text{ true} & k \neq \ell \\ |s| \text{ true} & k = \ell \end{cases} & \\
|J_1, \ldots, J_n| & = & |J_1|, \ldots, |J_n| &
\end{array}
$$

Sequents

$$
|\Sigma; \Gamma \xrightarrow{k} s \text{ true }| \quad = \quad \begin{cases} \Sigma; |\Gamma| \rightarrow k \text{ affirms } |s| & k \neq \ell, \mathsf{principal} \notin \Sigma \\ \Sigma; |\Gamma| \rightarrow |s| \text{ true} & k = \ell, \mathsf{principal} \notin \Sigma \end{cases}
$$

Figure 3.6: Translation from a fragment of BL_S to GP logic

To show: $\Sigma; |\Gamma|, p \rightarrow p$

This follows directly from rule (init).

Case. $\dfrac{\Sigma \vdash k \succeq k_0 \qquad \Sigma; \Gamma, k \text{ claims } s, s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{k_0} r}\text{claims } (k_0 \neq \ell)$

Subcase. $k = \ell$. To show: $\Sigma; |\Gamma|, |s| \rightarrow k_0 \text{ affirms } |r|$

1. $\Sigma; |\Gamma|, |s|, |s| \rightarrow k_0 \text{ affirms } |r|$ (i.h. on premise)
2. $\Sigma; |\Gamma|, |s| \rightarrow k_0 \text{ affirms } |r|$ (Contraction on 1)

Subcase. $k \neq \ell$. To show: $\Sigma; |\Gamma|, k \text{ says } |s| \rightarrow k_0 \text{ affirms } |r|$
By assumption, the premise $\Sigma \vdash k \succeq k_0$ implies $k = k_0$.

1. $\Sigma; |\Gamma|, k \text{ says } |s|, |s| \rightarrow k_0 \text{ affirms } |r|$ (i.h. on premise)
2. $\Sigma; |\Gamma|, k \text{ says } |s| \rightarrow k_0 \text{ affirms } |r|$ (Rule (saysL) on 1; $k = k_0$)

Case. $\dfrac{\Sigma \vdash k \succeq \ell \qquad \Sigma; \Gamma, k \text{ claims } s, s \xrightarrow{\ell} r}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{\ell} r}\text{claims}$

The premise $\Sigma \vdash k \succeq \ell$ forces $k = \ell$. Therefore, we must show that $\Sigma; |\Gamma|, |s| \to |r|$.

1. $\Sigma; |\Gamma|, |s|, |s| \to |r|$ (i.h. on premise)
2. $\Sigma; |\Gamma|, |s| \to |r|$ (Contraction on 1)

Case. $\dfrac{\Sigma; \Gamma| \xrightarrow{k} s}{\Sigma; \Gamma \xrightarrow{k_0} k \text{ says } s}\text{saysR } (k_0 \neq \ell)$

Subcase. $k = \ell$. To show: $\Sigma; |\Gamma| \to k_0 \text{ affirms } |s|$

1. $\Sigma; |(\Gamma|)| \to |s|$ (i.h. on premise)
2. $\Sigma; |\Gamma| \to |s|$ (Weakening on 1)
3. $\Sigma; |\Gamma| \to k_0 \text{ affirms } |s|$ (Rule (affirms) on 2)

Subcase. $k \neq \ell$. To show: $\Sigma; |\Gamma| \to k_0 \text{ affirms } k \text{ says } |s|$

1. $\Sigma; |(\Gamma|)| \to k \text{ affirms } |s|$ (i.h. on premise)
2. $\Sigma; |\Gamma| \to k \text{ affirms } |s|$ (Weakening on 1)
3. $\Sigma; |\Gamma| \to k \text{ says } |s|$ (Rule (saysR) on 2)
4. $\Sigma; |\Gamma| \to k_0 \text{ affirms } k \text{ says } |s|$ (Rule (affirms) on 3)

Case. $\dfrac{\Sigma; \Gamma| \xrightarrow{k} s}{\Sigma; \Gamma \xrightarrow{\ell} k \text{ says } s}\text{saysR}$

Subcase. $k = \ell$. To show: $\Sigma; |\Gamma| \to |s|$

1. $\Sigma; |(\Gamma|)| \to |s|$ (i.h. on premise)
2. $\Sigma; |\Gamma| \to |s|$ (Weakening on 1)

Subcase. $k \neq \ell$. To show: $\Sigma; |\Gamma| \to k \text{ says } |s|$

1. $\Sigma; |(\Gamma|)| \to k \text{ affirms } |s|$ (i.h. on premise)
2. $\Sigma; |\Gamma| \to k \text{ affirms } |s|$ (Weakening on 1)
3. $\Sigma; |\Gamma| \to k \text{ says } |s|$ (Rule (saysR) on 2)

Case. $\dfrac{\Sigma; \Gamma, k \text{ says } s, k \text{ claims } s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \text{ says } s \xrightarrow{k_0} r}\text{saysL}\ (k_0 \neq \ell)$

To show: $\Sigma; |\Gamma|, |k \text{ says } s| \longrightarrow k_0 \text{ affirms } |r|$

1. $\Sigma; |\Gamma|, |k \text{ says } s|, |k \text{ claims } s| \longrightarrow k_0 \text{ affirms } |r|$ (i.h. on premise)
2. $\Sigma; |\Gamma|, |k \text{ says } s| \longrightarrow k_0 \text{ affirms } |r|$ (Contraction on 1; $|k \text{ says } s| = |k \text{ claims } s|$)

Case. $\dfrac{\Sigma; \Gamma, k \text{ says } s, k \text{ claims } s \xrightarrow{\ell} r}{\Sigma; \Gamma, k \text{ says } s \xrightarrow{\ell} r}\text{saysL}$

To show: $\Sigma; |\Gamma|, |k \text{ says } s| \longrightarrow |r|$

1. $\Sigma; |\Gamma|, |k \text{ says } s|, |k \text{ claims } s| \longrightarrow |r|$ (i.h. on premise)
2. $\Sigma; |\Gamma|, |k \text{ says } s| \longrightarrow |r|$ (Contraction on 1; $|k \text{ says } s| = |k \text{ claims } s|$)

□

Theorem 3.19 (Completeness)**.** *If* $\ulcorner \Sigma; \Gamma \longrightarrow \gamma \urcorner$ *is provable in* BL_S*, then* $\Sigma; \Gamma \longrightarrow \gamma$ *is provable in GP logic.*

Proof. Suppose $\ulcorner \Sigma; \Gamma \longrightarrow \gamma \urcorner$ is provable in BL_S. By Lemma 3.18, $|\ulcorner \Sigma; \Gamma \longrightarrow \gamma \urcorner|$ is provable in GP logic. Using Lemma 3.17, $|\ulcorner \Sigma; \Gamma \longrightarrow \gamma \urcorner| = \Sigma; \Gamma \longrightarrow \gamma$. Hence $\Sigma; \Gamma \longrightarrow \gamma$ is provable in GP logic. □

3.5.2 Translation from Soutei

Soutei is a trust management system, i.e. a framework for administering and enforcing authorization policies [118]. It has been deployed in at least one large application, namely a publish-subscribe service on the web. Soutei's language for writing authorization policies is declarative; its syntax extends the syntax of Prolog by allowing limited use of the connective k says s. For the lack of a better name we call this language SL. The semantics of SL are defined through inference rules that resemble backchaining rules for top down logic programming (*a la* Prolog). An extremely interesting aspect from our perspective is that the says connective in SL behaves similarly to the says modality in BL_S to the extent that, with the exception of differences in syntax, SL is a fragment of BL_S. In the following we describe SL, and prove that it can be embedded in BL_S. A translation is needed to account for the differences in syntax. Although the inference system of SL allows the same consequences as that of BL_S, the two differ significantly in details. As a result, the proof of correctness of the embedding is quite involved.

SL. SL is based on Binder [52], another declarative language for writing authorization policies. Policy statements (called clauses) are divided into disjoint sets called *assertions*. Each assertion has a name, which is analogous to a principal in authorization logics. If $c_1, \ldots, c_n$ are the clauses in an assertion named k, then the whole assertion behaves as the hypothesis k says $c_1, \ldots, k$ says c_n. The syntax of SL is shown below.[6]

Principals or names	k		
Atomic Formulas	p	::=	$P\ t_1 \ldots t_n$
Goals	g	::=	$p \mid k$ says p
Clauses	c	::=	$\forall x_1 \ldots x_n.\ (p$:- $g_1, \ldots, g_m)$
Assertions	Δ	::=	$c_1, \ldots, c_n$
Named assertions	N	::=	$k : \Delta$
Hypotheses	Γ	::=	$N_1, \ldots, N_m$
Queries	q	::=	$\Delta \vdash_\Gamma g$

Policy statements are represented as clauses that have the form $\forall x_1 \ldots x_n.\ (p$:- $g_1, \ldots, g_m)$, where p is an atomic formula and each g_i is either an atomic formula or has the form k says p. As usual, the entire clause implies that for any grounding substitution θ with domain $x_1 \ldots x_n$, $p\theta$ holds if each of $g_1\theta, \ldots, g_m\theta$ hold. n may be zero, in which case $p\theta$ is a fact. An assertion Δ is a set of clauses. A *named assertion* is a pair $k : \Delta$ containing an assertion and a principal. The principal is a name for the assertion, and may represent a physical domain (such as a computer or a user) inside which policies contained in the assertion hold. The set of all named assertions is called the hypotheses Γ. It is assumed implicitly that the names of all assertions in Γ are distinct. Authorization queries are evaluated relative to the hypotheses Γ and an assertion Δ containing clauses which are valid at the point of evaluation. As evaluation of a query proceeds, Δ may change, but Γ remains fixed. Evaluation of queries is goal directed, and uses the following two rules:

$$\frac{(\forall x_1 \ldots x_n.\ (p \text{ :- } g_1, \ldots, g_m)) \in \Delta \qquad \mathtt{dom}(\theta) \supseteq x_1 \ldots x_n \qquad (\Delta \vdash_\Gamma g_i\theta)_{i \in \{1,\ldots,m\}}}{\Delta \vdash_\Gamma p\theta}\,\mathrm{bc}$$

$$\frac{(k : \Delta') \in \Gamma \qquad \Delta' \vdash_\Gamma p}{\Delta \vdash_\Gamma k \text{ says } p}\,\mathrm{says}$$

The rule (bc) means that $p\theta$ holds if there is a clause $\forall x_1 \ldots x_n.\ (p$:- $g_1, \ldots, g_m)$ in the valid assertion, and each $g_i\theta$ holds. This is the standard backchaining rule for logic programs. The rule (says) means that k says P is true if in the assertion Δ named k, p is true.

Translation from SL to $\mathbf{BL}_S$. Assume that BL_S contains all principals in SL, each of which is distinct from ℓ. Further assume that unequal principals of SL are not related to each other in the order $\succeq$. Since SL is not multi-sorted, we need only one sort in BL_S,

[6] We change Soutei's original notation to make it consistent with our own notation. We also simplify the evaluation rules slightly, without affecting their consequences.

Goals g

$$\begin{array}{rcl} \ulcorner p \urcorner & = & p \\ \ulcorner k \text{ says } p \urcorner & = & k \text{ says } p \end{array}$$

Clauses c

$$\ulcorner \forall x_1 \ldots x_n.\ (p \text{ :- } g_1, \ldots, g_m) \urcorner \quad = \quad \forall x_1 \ldots x_n.\ ((\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p)$$

Assertions Δ

$$\ulcorner c_1, \ldots, c_n \urcorner \quad = \quad \ulcorner c_1 \urcorner, \ldots, \ulcorner c_n \urcorner$$

Named assertions N

$$\ulcorner k : c_1, \ldots, c_n \urcorner \quad = \quad k \text{ claims } \ulcorner c_1 \urcorner, \ldots, k \text{ claims } \ulcorner c_n \urcorner$$

Hypotheses Γ

$$\ulcorner N_1, \ldots, N_m \urcorner \quad = \quad \ulcorner N_1 \urcorner, \ldots, \ulcorner N_m \urcorner$$

Figure 3.7: Translation from SL to BL_S

say principal. In this special case, $\Sigma \vdash t$: principal whenever all free variables of t are in the domain of Σ. The translation $\ulcorner \cdot \urcorner$ from SL to BL_S is defined in Figure 3.7. Since the only sort is principal, we omit sort annotations from universal quantifiers, abbreviating $\forall x{:}\text{principal}.s$ to $\forall x.s$.

It is noteworthy that the translation only renames some connectives. It replaces :- by $\supset$, and the named assertion $k : c_1 \ldots c_n$ by k claims $c_1, \ldots, k$ claims c_n. Owing to the similarity in the behavior of says in Soutei and BL_S, this simple translation is sound and complete. This is formalized by the following theorem.

Theorem 3.20 (Correctness)**.** *Suppose* $k : \Delta \in \Gamma$. *Then,* $\Delta \vdash_\Gamma g$ *in SL if and only if* $\cdot; \ulcorner \Gamma \urcorner, \ulcorner \Delta \urcorner \xrightarrow{k} \ulcorner g \urcorner$ *in* BL_S.

Proof. The "only if" direction follows by a simple induction on derivations in Soutei. The "if" direction is much harder to establish. Our proof follows an approach similar to that of Theorem 3.19 and relies on the subformula property. The inverse translation, however, is more involved. Details of the proof in both direction are in Appendix A (Theorem A.11). □

3.6 Horn Fragment and Translation to First-Order Logic

As the final technical result of this chapter we present a sound and complete embedding of a reasonably expressive fragment of BL_S in first-order intuitionistic logic.[7] The main idea of our translation is to eliminate the modality k says s by pushing the principal name k to the predicates in s as an extra argument. For example, we translate k says $(P\ t_1 \dots t_n)$ to $(P\ k\ t_1 \dots t_n)$. Besides the fact that the translation makes BL_S amenable to existing tools for first-order logic such as automatic theorem provers, the translation is also relevant for a historic reason – its main idea has been used in the past to both define and implement other declarative policy languages with the says modality. For example, the semantics of Binder, one of the earliest policy languages with a says modality, are defined using a similar translation that maps Binder policies into Datalog programs [52]. Similarly, the policy language SecPAL is implemented via translation to Datalog, again using a related translation to embed says in first-order logic [23]. What the results of this section show is that there is at least some logical justification for these translations, namely that for a says modality that behaves like the one in BL_S, pushing k says $\cdot$ to predicates constitutes a provability preserving embedding.

We make two important observations. First, this translation does not work for all existing authorization logics with a says modality. The translation is not sound if the says modality in the source logic is too strong, e.g., as in the GP logic (§3.5.1). Similarly, the translation is not complete if the says modality is too weak, as happens, for example, with logics in early work [8, 88]. Therefore, the interpretation of says should neither be too weak nor too strong for the translation to be sound and complete. BL_S seems to achieve this balance well. Second, even for BL_S, the use of connectives must be restricted in source formulas in order to make the translation complete. More precisely, what we really translate is not all of BL_S but only a fragment of it. This fragment is quite large and very expressive. Indeed, all policies encountered by the author so far that can be expressed in BL_S can also be expressed in the fragment, and the image of the translation from SL (§3.5.2) is also contained in it. The main restriction in the fragment is that $\supset$ and $\forall$ are not allowed to appear as top level connectives in conclusions of sequents, whereas $\exists$, $\vee$, and $\bot$ are not allowed to appear at the top level in hypotheses. Further, it must be assumed that the order $\succeq$ between principals is trivial, i.e. $\Sigma \vdash k \succeq k'$ implies $k = k'$. The fragment resembles a fragment of predicate logic called the Horn fragment, and hence we call it the Horn fragment of BL_S.[8] An important property is that in any proof of a sequent in the Horn fragment, the sorting Σ and the hypotheses Γ never change.

Horn fragment of BL_S. The syntax of the Horn fragment of BL_S is shown below. We divide the syntax of formulas into goals g and clauses d (the terms goal and clause are borrowed from logic programming). Hypotheses are restricted to the forms k claims d and

[7] We drop the adjective "intuitionistic" when referring to the target of the translation since the image of the translation lies in a fragment of first-order logic on which intuitionistic and classical provability coincide. We do not show here that this is the case, since this observation is orthogonal to the concerns of the translation. Our correctness proof is based on an intuitionistic sequent calculus.

[8] The Horn fragment of predicate logic is the fragment that is used in Prolog.

d true (abbreviated to d). The two forms of hypotheses are distinguished using different letters Δ and Ξ respectively. This is necessary because the translations of the two types of hypotheses are different. In addition we assume that distinct principals are unrelated to each other in the order $\succeq$ (so $\succeq$ is the diagonal relationship); in particular, ℓ is assumed to be absent.

Goals	g	::=	$p \mid k$ says $g \mid g_1 \wedge g_2 \mid g_1 \vee g_2 \mid \top \mid \bot \mid \exists x{:}\sigma.g$
Clauses	d	::=	$p \mid \forall x{:}\sigma.d \mid g \supset d \mid \top \mid d_1 \wedge d_2$
Claims Hypotheses	Δ	::=	k_1 claims $d_1, \ldots, k_n$ claims d_n
True Hypotheses	Ξ	::=	$d_1, \ldots, d_n$
Sequents			$\Sigma; \Delta, \Xi \xrightarrow{k} g$

The rules of inference for the Horn fragment are the same as those of the sequent calculus for BL_S (Figure 3.3), except that sequents are restricted to the form $\Sigma; \Delta, \Xi \xrightarrow{k} g$. It can be checked that this class of sequents is closed in the following sense: all sequents occurring in the proof of a sequent in the class also lie in the class.

Translation to first-order logic. As the target of the translation we consider a multi-sorted first-order intuitionistic logic having the same sorts as BL_S. We assume that for every predicate in BL_S, there is a predicate of the same name in first-order logic that takes an extra argument of sort $\mathsf{principal}$. As a convention, we make this the first argument of the predicate. The proof theory of intuitionistic first-order logic has been studied extensively and although we need it for proving the translation correct, we do not reiterate it here. Briefly, a sequent calculus for the logic may be obtained by ignoring rules containing says and claims in Figure 3.3 and additionally dropping all views from sequents (see Figure A.1 in Appendix A for a listing of the rules of the sequent calculus).

Figure 3.8 describes the translation $[\![\cdot]\!]$ from the Horn fragment to first-order logic. An auxiliary translation $[\![\cdot]\!]_k$ indexed by a principal k is also needed to translate formulas (goals and clauses) and true hypotheses Ξ. Intuitively, the index k is the principal in the nearest says or claims outside the formula being translated. The central "trick" of the translation is to push the modality k says $\cdot$ down to atomic formulas, where k is added as an extra argument to the predicate symbols. That this simple idea works for a reasonably large fragment of BL_S may seem surprising. However, as mentioned earlier, the idea does not extend to larger fragments.

Theorem 3.21 (Correctness of Translation)**.** *Let $\Sigma; \Delta, \Xi \xrightarrow{k} g$ be a sequent in the Horn fragment of BL_S and assume that for each $d \in \Xi$, k claims $d \in \Delta$. Then $\Sigma; \Delta, \Xi \xrightarrow{k} g$ is provable in BL_S if and only if its translation $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$ is provable in first-order logic.*

Proof. Soundness, the "only if" direction, follows by a straightforward induction on the proof of $\Sigma; \Delta, \Xi \xrightarrow{k} g$. Completeness follows by a lexicographic induction, first on the given derivation of $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$ and then on the structure of g. Details of the proof in both directions are in Appendix A (Theorem A.14). □

Goals g

$$
\begin{array}{rcl}
[\![P\ t_1 \ldots t_n]\!]_k & = & P\ k\ t_1 \ldots t_n \\
[\![k' \ \mathsf{says}\ g]\!]_k & = & [\![g]\!]_{k'} \\
[\![g_1 \wedge g_2]\!]_k & = & [\![g_1]\!]_k \wedge [\![g_2]\!]_k \\
[\![g_1 \vee g_2]\!]_k & = & [\![g_1]\!]_k \vee [\![g_2]\!]_k \\
[\![\top]\!]_k & = & \top \\
[\![\bot]\!]_k & = & \bot \\
[\![\exists x{:}\sigma.g]\!]_k & = & \exists x{:}\sigma.[\![g]\!]_k
\end{array}
$$

Clauses d

$$
\begin{array}{rcl}
[\![P\ t_1 \ldots t_n]\!]_k & = & P\ k\ t_1 \ldots t_n \\
[\![\forall x{:}\sigma.d]\!]_k & = & \forall x{:}\sigma.[\![d]\!]_k \\
[\![g \supset d]\!]_k & = & [\![g]\!]_k \supset [\![d]\!]_k \\
[\![\top]\!]_k & = & \top \\
[\![d_1 \wedge d_2]\!]_k & = & [\![d_1]\!]_k \wedge [\![d_2]\!]_k
\end{array}
$$

Claims Hypotheses Δ

$$
[\![k_1 \ \mathsf{claims}\ d_1, \ldots, k_n \ \mathsf{claims}\ d_n]\!] = [\![d_1]\!]_{k_1}, \ldots, [\![d_n]\!]_{k_n}
$$

True Hypotheses Ξ

$$
[\![d_1, \ldots, d_n]\!]_k = [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k
$$

Sequents

$$
[\![\Sigma; \Delta, \Xi \xrightarrow{k} g]\!] = \Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \rightarrow [\![g]\!]_k
$$

Figure 3.8: Translation from the Horn fragment of BL_S to first-order logic.

Example 3.22. We illustrate the translation $[\![\cdot]\!]$ on the policy of Examples 3.1 and 3.6. The policy in Figure 3.1 does not lie in the Horn fragment (because clauses cannot contain says at the top level), but that policy can be transformed to an equivalent one by replacing all top level says with claims. This slightly modified policy, which we denoted with the symbol Γ' in Example 3.6, is in the syntax of claims hypotheses (Δ). We may translate the policy using the translation $[\![\cdot]\!]$. As illustrations, rules (1) and (2) when translated result in the following formulas.

$$(1') \quad \forall k, k', f.\ (((\texttt{hasLevelForFile}\ \mathsf{admin}\ k\ f) \wedge (\texttt{owns}\ \mathsf{system}\ k'\ f) \wedge (\texttt{may}\ k'\ k\ f\ \mathsf{read})) \supset \texttt{may}\ \mathsf{admin}\ k\ f\ \mathsf{read})$$

$$(2') \quad \forall k, f, l, l'.\ (((\texttt{levelFile}\ \mathsf{system}\ f\ l) \wedge (\texttt{levelPrin}\ \mathsf{hr}\ k\ l') \wedge (\texttt{below}\ \mathsf{admin}\ l\ l')) \supset \texttt{hasLevelForFile}\ \mathsf{admin}\ k\ f)$$

Since the translation does not account for the relation $\succeq$, the rules (3)–(5) from Figure 3.1 that were stated by ℓ must be replicated for every principal in first-order logic. For instance, instead of assuming the formula $\texttt{below}\ \ell\ \mathsf{secret}\ \mathsf{topsecret}$ (which would be the translation of (4)), we need to assume $\texttt{below}\ \mathsf{admin}\ \mathsf{secret}\ \mathsf{topsecret}$ to draw meaningful conclusions from the translated policy.

Another relevant observation is that although Theorem 3.21 shows that translated policies can be used to draw the same authorizations as the original policies in BL_S, translated policies are not very convenient for direct enforcement in distributed settings. The reason is that it is not obvious from either (1') or (2') that they correspond to rules of the principal admin, and consequently, it is also unclear how they may be established to a reference monitor. On the other hand, the corresponding rules (1) and (2) from Figure 3.1 make explicit the identity of the principal creating them (via the top level annotation $\mathsf{admin}\ \mathsf{says}\ \cdot$), and make it obvious that both (1) and (2) should be established through certificates signed by admin. Similarly, if first-order logic were to be used for enforcement, all certificates would have to be represented via the translation. As a result, the translation described in this section is largely of theoretical interest. It can used in practice indirectly, e.g., for reducing the problem of proof search in BL_S to that of proof search in first-order logic.

A formal relation between Binder and Soutei. The creators of Soutei emphasize (without proof) [118, Section 2] that Soutei is a dialect of another policy language, Binder [52]. Whereas the syntax of Soutei is a restriction of the syntax of Binder, it is far from obvious that their seemingly different inference systems admit the same authorizations from syntactically identical policies. As explained in §3.5.2, the inference system of Soutei resembles a backchaining proof system from logic programming. The semantics of Binder policies, on the other hand, are defined via a translation to first-order logic which is identical to the translation described above. Consequently, Theorems 3.20 and 3.21 together show that the authorizations derivable from policies expressible in Soutei are the same in Binder and Soutei (and also in BL_S). Therefore, Soutei is *provably* a fragment of Binder.

3.7 Related Work

We close this chapter with a discussion of some of the vast amount of work on other authorization logics, logic-based authorization languages, and other formalisms for expressing authorization policies. We also discuss work on formal analysis of policies for correctness and problems, a topic that is not covered in this thesis. Related work on constructs in policy formalisms that may be used to express dynamic policies is discussed in §4.7.

3.7.1 Authorization Logics

The ABLP logic. The study of authorization logics was initiated in the work of Lampson and others [8, 88]. The logic proposed in these papers, called the ABLP logic, is classical and its proof system is axiomatic (as opposed to BL_S and the GP logic which are intuitionistic and based in structural proof theory). The main goal of the ABLP logic was to formalize and *explain* authentication and authorization in distributed systems; the logic was not intended for a direct implementation.

The ABLP logic introduced the modality says although the behavior of the modality is different from says in BL_S because the modality satisfies the rule (N) and axiom (K) from §3.1, but no other primitive axioms. As a result, k says $((k' \ \mathsf{says}\ s') \supset s)$ and k' says s' do not imply k says s in the logic, which makes it impossible to express authorizations in the manner considered in this chapter. The logic is propositional, but it contains the speaksfor connective $k_1 \Rightarrow k_2$ (see the discussion of full delegation in §3.1.2). As discussed in §3.1.2, the order $\succeq$ on principals in BL_S can be used to fulfill the same purpose as speaksfor, although BL_S suffers from the limitation that $\succeq$ is not internalized into the syntax of formulas. The latter restriction is removed in the full logic BL in §4.

Another important aspect of the ABLP logic, also not present in BL_S, are principals with a syntactic structure that has semantic consequences. For example, the principal $k_1 \wedge k_2$ has the property that for every s, $(k_1 \wedge k_2)$ says s is logically equivalent to $(k_1$ says $s) \wedge (k_2$ says $s)$. Many other "connectives" of principals besides $\wedge$ are considered in the ABLP logic. It is argued, mostly informally, that such structured principals can be used to represent several policy idioms including many kinds of delegations, groups, and roles [8]. A significant technical result of the work relates to decidability of fragments of the logic.

The logic of Appel and Felten. In their seminal work on proof-carrying authorization (then called proof-carrying *authentication*), Appel and Felten introduced a higher-order authorization logic [13]. In this logic, principals are treated as predicates and k says s is defined as $k(s)$ (predicate k applied to formula s). The deduction system contains some common inference rules of higher-order logic and some specialized rules for authorization specific concerns. Higher-order predicates make the logic very expressive; indeed the authors treat the logic as a logical framework in which other authorization logics may be encoded. However, owing to the higher-order constructs, even proving simple properties like consistency is extremely difficult. Further, it remains unclear why a higher-order logic is necessary when first-order quantification suffices not only to express most policy idioms, but also as the basis of extremely expressive logical frameworks like Twelf [116]. This logic

and its derivatives form the basis of several implementations of proof-carrying authorization prior to this work [18, 20].

GP logic and related approaches. As discussed in §3.5.1, the GP logic was developed jointly by the author and Pfenning [67]. The says connective in the GP logic is treated as an indexed lax modality [28, 60]. The paper introduces three basic ideas: (a) use of intuitionistic logic as opposed to classical logic for expressing authorization policies, (b) emphasis on structural proof theory and metatheory for authorization logic, and (c) use of first-order quantifiers in place of higher-order quantifiers as in the work of Appel and Felten. (a) and (b) are foundationally important contributions. They were discussed in §1. (c), the use of first-order quantifiers, was already "in the air" at the time that the GP logic was conceived because it was present in several policy languages (described below). Through a translation from the GP logic to BL_S, it was argued in §3.5.1 that BL_S is no less expressive than the GP logic. Further, it has been described in §3.1.2 that it is difficult to express exclusive delegation in the GP logic. The latter motivated the switch from the GP logic to BL as the basis of this thesis. The ideas of using intuitionistic connectives, structural proof theory, and first-order quantifiers as foundations for authorization carry over to BL.

In independent work, Abadi describes a logic closely related to the GP logic [5], derived as a special case of the dependency core calculus (DCC) [7]. DCC treats k says s as an indexed lax modality (like the GP logic), but it allows more. For example, k says k' says s is logically equivalent to k' says k says s. This can be undesirable in many scenarios. A restricted version of DCC, called CDD is very similar to the GP logic (in particular it does not admit this commutativity), except that it contains second-order quantification in place of first-order quantification. While this makes it difficult to express many policies that require quantification over principals and objects, it does allow the speaksfor connective introduced in §3.1.2 to be defined using other connectives.

In joint work with Abadi, we consider many extensions of the propositional fragment of the GP logic (and CDD) under the name ICL [65]. In particular, structured principals from ABLP logic are revisited, and their semantics are precisely defined. Further, the nature of the indexed lax modalities is explained by translation to the modal logic S4. It is also shown that the interpretation of $k \Rightarrow k'$ as $\forall s.\ ((k \text{ says } s) \supset (k' \text{ says } s))$ is sound and complete if certain conditions are met.

The GP logic and CDD have been used subsequently in many other places, including languages for security [15, 61, 85, 139], several extensions including those containing support for explicit time and consumable credentials [34, 54, 66], an extensible authorization framework [90], and an extended logic for both representing authorization policies and reasoning about their consequences [55].

Other work. In a survey of the use of logic in access control, Abadi explores connections between logics and languages for writing authorization policies, in particular, Binder [4]. In this context, he proposes the axiom (I), which is admissible in BL_S. More recently, Abadi has studied possible axioms for authorization logics and the connections between them, both in classical and intuitionistic settings [6].

3.7.2 Logic-based Authorization Languages

Besides authorization logics, there are several languages for writing authorization policies and determining their consequences that use logical syntax and logic-like inference rules. In this section we discuss some of these languages.

Historically, Binder [52] was the first policy language to support distributed policies. Its syntax extends the Horn fragment of predicate logic with a says modality, and its semantics (consequence relation) are defined by a translation to first-order logic, as discussed in §3.6. The overall structure of policy evaluation is similar to proof-carrying authorization: principals may sign arbitrary policy statements, which are then distributed to others. Any principal may derive authorizations by translating all policy statements it has into first-order logic using a transformation similar to that in §3.6 and running a Datalog engine over the translated statements. Soutei is another policy language with a says modality [118]. What is interesting about Soutei here is the similarity between its says modality and that of BL_S – we proved in §3.5.2 that Soutei is a fragment of BL_S. Soutei is also closely connected to, and a fragment of Binder, as argued in §3.6.

SecPAL [23] is a more recent language for writing authorization policies. In addition to the says modality, SecPAL also includes support for exclusive delegation (via a construct written k says k' `cansay` p) as well as limited support for bounded delegation (a delegation in SecPAL can either be undelegatable or it can be delegated to any depth). SecPAL also supports environmental constraints including the a limited form of explicit time; this aspect is discussed in greater detail in §4.7. SecPAL's formal semantics are based on logic-like inference rules, while its implementation is based on a translation to Datalog. It is shown formally that the translation respects the inference rules. Recently, Dinesh et al. [56] have proposed an access control logic, which generalizes the `cansay` construct of SecPAL from atomic to arbitrary formulas. The proof system of the logic is classical and axiomatic. The language DKAL [76] adds directed communication to SecPAL. With directed communication, principals may make statements that are heard only by intended recipients. DKAL combines authorization policies and logic-based framework for reasoning about knowledge of principals. The latter is also the subject of a recent paper on an extension of the GP logic called ω-logic [55].

The policy language RT [95] combines role based access control (RBAC) with trust management. We include it here because its semantics are defined through a translation to an extension of Datalog with constraint domains called Constrained Datalog [94]. RT has many important constructs including a construct to encode separation of duty as well as a thresholding construct that allows authorization only when m out of n designated principals approve. The latter seem to be difficult to express in authorization logics without additional constraint domains due to exponential blow-up in encoding (See §4.1.2 for a description of how thresholding can be encoded with constraints in the full logic BL). Cassandra [26] is another policy language based on RBAC whose semantics are also defined by translation to Datalog with constraints. In addition to roles, Cassandra also has support for representing physical distribution of policies on different sites. Other similar languages include DL whose semantics are defined by translation to logic programs [92], and SD3 which is a very simple extension of Datalog with a certified inference engine that creates and checks a proof of

inference before returning the result [86].

3.7.3 Other Policy Formalisms

There are also many other formalisms for expressing authorization policies and reasoning from them that are not based in logic. Although they are not directly connected to the work of this thesis, for the sake of completeness, we briefly discuss some of them here.

Trust Management. Trust management (TM) is a general term for describing management of delegation in authorization policies. It was coined by Blaze et al. [32] who introduced two frameworks for enforcing policies, PolicyMaker [33] and its successor KeyNote [31]. The basic construct in these frameworks is delegation: principals delegate authority over specific subjects (predicates) to others through digitally signed certificates. For example, Alice may delegate Bob the authority to make decisions about access to e-mails.

Although designed as certificate schemes for binding names and keys to principals, the Simple Public Key Infrastructure (SPKI) [58] and X.509v3 [79] also allow delegation of authority in a manner similar to TM frameworks. SPKI also allows limited control over delegating delegated authority (bounded delegation; §3.1.2). When giving Bob some authority, Alice may or may not allow Bob to further delegate the authority. SPKI and KeyNote also allow authority to be delegated jointly to groups of principals, and like RT, also support thresholding. The latter means that an authorization holds if at least m distinct principals out of n specified principals state that it does.

Role Based Access Control (RBAC). RBAC [127] is a generic approach to access control in which permissions are authorized to specific roles, and principals are assigned membership to roles based on need. A lot of work has been done in the area, including a language for enforcing RBAC policies [83], and several proposals for administering RBAC [93, 126, 135].

XACML. XACML [107] is an XML based language for specifying policies. XACML can express attribute based authorization. Because XACML policy rules may explicitly allow or deny access, decisions drawn from policies may be inconclusive in some cases.

3.7.4 Policy Analysis

There has been a limited amount of work in the past on analyzing formally represented authorization policies for desirable and undesirable properties and checking their correctness against intuitive criteria. Techniques in the area have been based in logic as well as other formal methods.

Proof-theoretic approaches to analysis of authorization policies were pioneered in joint work of the author and Pfenning [67]. That work describes a method for making a static approximation of an authorization policy written in GP logic to prove that the addition of certain kinds of formulas (credentials) cannot not affect the consequences drawn from the policy. This can be used to prove that the policy satisfies some intuitive properties

regarding what control specific principals have over specific predicates. It is also proved that a priori, the logic isolates the statements of principals. This is called non-interference. It is a metatheorem like admissibility of cut. A similar (in fact stronger) theorem can be proved for BL_S easily. Abadi describes a related but different notion of non-interference for DCC [5]. Chaudhuri et al. describe a Datalog-based framework for modeling and analyzing the consequences of access control systems [42]. They apply their method to the access control model of Windows Vista and the Asbestos operating system, and find vulnerabilities in the former.

Outside of logic, there has been work on analysis of RBAC systems. For example, Li and Mitchell describe algorithms for analyzing reachability properties for states of an RBAC system, given some restrictions on how the policies may change [96]. Schaad and Moffett [131] present a specification language for describing conflict-free role based systems. Sasturkar et al. [129] and Stoller et al. [135] analyze the complexity of deciding reachability of states in RBAC systems with administrators (ARBAC).

Chapter 4

BL: An Authorization Logic for Dynamic Policies

This chapter introduces BL, the logic used in PCFS, discusses its proof theory, and its metatheoretic properties. BL is an extension of BL_S (§3); from the latter it inherits all connectives of first-order intuitionistic logic and the modality k says s. In addition, BL supports explicit time, predicates that represent the state of the system (called interpreted predicates), and constraint domains. Using these constructs, dynamic policies that depend on time and system state can be expressed in BL (§1.3, Motivation 2).

Support for explicit time in BL is manifest in the modality s @ $[u_1, u_2]$, which means that s is always true during the time interval $[u_1, u_2]$ but possibly not outside of it. u_1 and u_2 both denote time points, encoded as integers that count seconds from a fixed point of reference. This modality is useful for representing policies that expire at stipulated points of time, as well as those that use time relatively, e.g., allowing access for 90 days from the happening of an event. s @ $[u_1, u_2]$ was first studied in joint work by DeYoung, the author, and Pfenning [54] in the context of a different logic called η. η logic is covered in detail in DeYoung's undergraduate thesis [53]. It is an extension of the GP logic (§3.5.1) in much the same way that BL is an extension of BL_S. The proof-theoretic treatment of s @ $[u_1, u_2]$ in BL is largely based on that in η, with the exceptions that the interaction between k says s and explicit time in BL is subtler than it is in η, and that there is also a new interaction between explicit time and interpreted predicates in BL, which is absent from η since η does not include interpreted predicates. From the perspective of modal logics, s @ $[u_1, u_2]$ is a hybrid modality, and like other similar modalities it interacts with all connectives of the logic and changes the very nature of logical judgments [35, 106, 122]. This is discussed in detail in §4.2.

Explicit time is useful for determining consequences of policies in practice only in conjunction with methods for reasoning about inequality between time points. For example, if Alice has a certificate that allows her a certain access from January 01, 2007 to December 31, 2010, it is only reasonable that she be able to derive from it a proof that allows her access on August 12, 2009. Constructing such a proof requires the ability to reason that August 12, 2009 lies between January 01, 2007 and December 31, 2010. To this end, BL

includes special formulas called *constraints*, one particular form which may be inequalities on time points, $u_1 \leq u_2$. Constraints differ from ordinary predicates in that they are not established by hypothesis; instead their verification relies on an external constraint solver which is formally embedded in the logic via a satisfaction judgment $\models c$ (c denotes a constraint). We do not stipulate the rules of this judgment since, by design, the logic is agnostic to the solver used to implement constraints. However, the metatheoretic properties of BL are contingent upon certain assumptions about the constraint domain, which we list explicitly in §4.2.1. In addition to representing inequalities between time points constraints can be used to represent the relation $k \succeq k'$ between principals in BL_S, thus eliminating the need to fix the relation statically through the judgment $\Sigma \vdash k \succeq k'$. The proof-theoretic treatment of constraints in BL is based on similar work for linear logic [84, 128] which was also used previously in η logic.

Besides explicit time, many real authorization policies use the state of the system as an input. The state may represent the progress of a workflow or a protocol. As a simple example, the authorization policy for a homework directory in a class administration system may allow read and write access for the teaching assistants while the homework is being prepared, read and write access for students while the homework may be submitted, and read access for teaching assistants after submissions are closed. A simple way to model the different stages – preparation, submission, and post-submission – may be as a state system; the stage may be written by the instructor as an attribute (meta-data) on the homework directory, and the access policy rules may be contingent upon the value of the attribute. To incorporate such elements of state in policy rules, BL allows interpreted predicates, whose truth is not justified by the logical hypotheses, but by an external solver that refers to the state of the system. In the case of our example here, the solver would check the value of the attribute on the homework directory.

Constraints and interpreted predicates are similar to each other because both may be established through decision procedures external to the logic. They are different because the truth of interpreted predicates depends on the state of the system and may change with it, whereas the truth of constraints is independent of system state. This difference also manifests in separate treatments of constraints and interpreted predicates during enforcement of policies in PCFS (§5) and, hence, we maintain a syntactic distinction between the two in BL formulas. The proof theory of BL treats constraints and interpreted predicates similarly.

The emphasis in our discussion of BL, as in the case of BL_S, is on structural proof theory, i.e. a natural deduction system (§4.2.2) and a sequent calculus (§4.2.4), which we show to be equivalent in terms of provability (§4.2.6). We prove several metatheoretic properties for BL including admissibility of cut. The importance of structural proof theory and metatheory in the context of authorization have already been emphasized in §1.3 and §3, so we do not reiterate them here. However, it is perhaps useful to observe that besides the fact that proof theory defines the meanings of policies represented in BL, proof-theoretic techniques are directly implemented in the proof verifier and automatic prover for PCFS (§5 and §6). We do not consider an axiomatic system for BL; the author is uncertain if there even is a complete axiomatization of the logic.

An important aspect of structural proof theory that we study for BL is proof normalization (§4.5): we show that every natural deduction proof can be transformed to one in a restricted class of proofs called *canonical proofs*. Canonical proofs are proofs without any β-redexes.[1] The proof of this normalization result uses admissibility of cut and generalizes well known observations about the similarity of cut-elimination and proof normalization in classical and intuitionistic logic [119, 120, 145]. It is more directly based on notes on proof theory for intuitionistic logic by Pfenning [114]. Another interesting aspect of BL covered in this chapter is its connection to BL_S. Although BL's syntax and proof systems generalize those of BL_S, BL is not a conservative extension of BL_S. In particular, axiom (C) from §3.1.1 is not admissible in BL. In §4.6 we define a simple translation from BL_S to BL and prove it sound and complete. Finally, this chapter discusses how BL in used to represent policies and establish authorizations in PCFS (§4.3).

4.1 BL: Syntax and Informal Description

BL generalizes intuitionistic first-order logic with the connectives k says s and s @ $[u_1, u_2]$, constraints, and interpreted predicates. In addition to the sort principal already present in BL_S, BL also includes an additional sort time that includes all time points, whose members are denoted by the letter u. A ground time point is either an integer that represents time in seconds elapsed from a fixed point of reference (so time points can be both negative and positive), or it is one of the distinguished constants $-\infty$ and $+\infty$ denoting the minimum and maximum possible time respectively.

Atomic formulas in BL are of three types: (a) Uninterpreted atoms, p, which are obtained by applying uninterpreted predicates P to terms, (b) Interpreted atoms, i, which capture the state of the system in the logic, and (c) Constraints, c. BL does not stipulate any specific uninterpreted or interpreted predicates, but it requires at least two types of constraints: $k_1 \succeq k_2$, which represent the preorder on principals introduced in §3.1, and $u_1 \leq u_2$ that capture the usual ordering on integers with the added proviso that $u \leq +\infty$ and $-\infty \leq u$ for every u. The syntax of BL formulas is summarized in Figure 4.1. An interval $[u_1, u_2]$ in the syntax is well-formed only if $u_1 \leq u_2$. Well-formedness of formulas can be defined through inference rules as in prior work [54], but for simplicity we omit these details from the presentation here.

4.1.1 Properties of Connectives Explained Informally

Before describing proof systems for BL, we explain informally how explicit time, interpreted predicates, and constraints interact with other connectives of the logic and with each other. The objective of explaining these interactions is to illustrate the meanings of the new constructs in BL. Interaction of the modality k says s with standard connectives of first-order logic was explained in §3 in the context of BL_S and carries over to BL rather unchanged, the only exception being that the BL_S axiom (C) – k says $((k \text{ says } s) \supset s)$ – is not admissible

[1] A β-redex is any locus in a natural deduction proof where an elimination rule is applied to a connective that is established using an introduction rule.

Sorts	σ	::=	$\mathsf{principal} \mid \mathsf{time} \mid \ldots$
Integers	n	::=	$\ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots$
Constants	a	::=	$\ell \mid n \mid -\infty \mid +\infty \mid \ldots$
Terms	t, k, u	::=	$a \mid x \mid f(t_1, \ldots, t_n)$
Uninterpreted predicates	P		
Interpreted predicates	I		
Uninterpreted atoms	p, q	::=	$P\ t_1 \ldots t_n$
Interpreted atoms	i	::=	$I\ t_1 \ldots t_n$
Constraints	c	::=	$u_1 \leq u_2 \mid k_1 \succeq k_2 \mid \ldots$
Formulas	r, s	::=	$p \mid i \mid c \mid r \wedge s \mid r \vee s \mid r \supset s \mid \top \mid \bot \mid$ $\forall x{:}\sigma.s \mid \exists x{:}\sigma.s \mid k\ \mathsf{says}\ s \mid s\ @\ [u_1, u_2]$

Figure 4.1: Syntax of BL formulas

in BL. The latter is not a limitation of BL, since axiom (C) is rarely, if ever, needed to draw meaningful consequences from policies. It was included in BL_S primarily to make the axiomatic system and natural deduction equivalent (Theorem 3.13). An analogue of axiom (C), $(k\ \mathsf{says}\ (((k\ \mathsf{says}\ s)\ @\ [u_1, u_2]) \supset s))\ @\ [u_1, u_2]$ is admissible in BL.

Interaction of explicit time and constraints with other connectives. The BL modality $s\ @\ [u_1, u_2]$, together with constraints, interacts with all other connectives in a significant manner. Writing $s_1 \equiv s_2$ as an abbreviation for $(s_1 \supset s_2) \wedge (s_2 \supset s_1)$, and $\vdash s$ for provability without hypotheses, the following properties hold. A formal definition of $\vdash s$ and proofs of all properties listed in this section are deferred to §4.2.4.

1. $\vdash ((u_1 \leq u_1') \wedge (u_2' \leq u_2)) \supset ((s\ @\ [u_1, u_2]) \supset (s\ @\ [u_1', u_2']))$
2. $\vdash ((s_1 \wedge s_2)\ @\ [u_1, u_2]) \equiv ((s_1\ @\ [u_1, u_2]) \wedge (s_2\ @\ [u_1, u_2]))$
3. $\vdash ((s_1 \vee s_2)\ @\ [u_1, u_2]) \equiv ((s_1\ @\ [u_1, u_2]) \vee (s_2\ @\ [u_1, u_2]))$
4. $\vdash ((\forall x{:}\sigma.s)\ @\ [u_1, u_2]) \equiv (\forall x{:}\sigma.(s\ @\ [u_1, u_2]))$ $\quad (x \notin u_1, u_2)$
5. $\vdash ((\exists x{:}\sigma.s)\ @\ [u_1, u_2]) \equiv (\exists x{:}\sigma.(s\ @\ [u_1, u_2]))$ $\quad (x \notin u_1, u_2)$
6. $\vdash \top\ @\ [u_1, u_2]$
7. $\vdash (\bot\ @\ [u_1, u_2]) \supset (s\ @\ [u_1', u_2'])$
8. There is no interval $[u_1, u_2]$ such that $\vdash \bot\ @\ [u_1, u_2]$.
9. $\vdash ((s_1 \supset s_2)\ @\ [u_1, u_2]) \equiv (\forall x_1{:}\mathsf{time}.\forall x_2{:}\mathsf{time}.\ (((u_1 \leq x_1) \wedge (x_2 \leq u_2) \wedge (s_1\ @\ [x_1, x_2])) \supset (s_2\ @\ [x_1, x_2])))$

Property (1) means that if s holds during an interval $[u_1, u_2]$, then it also holds during any subinterval $[u_1', u_2']$. The constraints $u_1 \leq u_1'$ and $u_2' \leq u_2$ imply that $[u_1', u_2'] \subseteq [u_1, u_2]$.

Properties (2)–(5) mean that the @ connective commutes with the connectives $\wedge$, $\vee$, $\forall$, and $\exists$. The provability of the formula $((s_1 \vee s_2) \ @ \ [u_1, u_2]) \supset ((s_1 \ @ \ [u_1, u_2]) \vee (s_2 \ @ \ [u_1, u_2]))$ entailed by property (3) may be surprising. For example, if s_1 holds on some interval $[u_1, u]$ and s_2 holds on another interval $[u, u_2]$, then, seemingly, $(s_1 \vee s_2) \ @ \ [u_1, u_2]$ should be true but neither of $s_1 \ @ \ [u_1, u_2]$ and $s_2 \ @ \ [u_1, u_2]$ may hold. However, in BL, we do not allow an analysis of intervals; in particular, $s_1 \ @ \ [u_1, u]$ and $s_2 \ @ \ [u, u_2]$ do not imply $(s_1 \vee s_2) \ @ \ [u_1, u_2]$, so the previous counterexample does not work. The reasons for this choice are explained in §4.2.2.

Truth is provable a priori on all intervals as property (6) states. Property (7) states that if falsehood is provable on any interval, then every formula is provable on every interval. This may be surprising, particularly because a similar property does not hold for says – it is not the case that $\vdash (k \ \mathsf{says} \ \bot) \supset (k' \ \mathsf{says} \ s)$ for unrelated k and k'. However, there is no interval on which $\bot$ is provable a priori, so the logic is consistent (8). Property (9) means that a proof of $(s_1 \supset s_2) \ @ \ [u_1, u_2]$ is equivalent to having a proof of $(s_1 \ @ \ [x_1, x_2]) \supset (s_2 \ @ \ [x_1, x_2])$ for every subinterval $[x_1, x_2]$ of $[u_1, u_2]$. This property is a consequence of the intuitionistic nature of BL and the hybrid nature of @.

Unlike the connectives $\wedge$, $\vee$, $\forall$, and $\exists$ which commute freely with @, the says connective commutes with @ in only one direction, as the following properties show. ($\not\vdash s$ means that there is at least one instance of s that is not provable a priori.)

10. $\vdash ((k \ \mathsf{says} \ s) \ @ \ [u_1, u_2]) \supset (k \ \mathsf{says} \ (s \ @ \ [u_1, u_2]))$

11. $\not\vdash (k \ \mathsf{says} \ (s \ @ \ [u_1, u_2])) \supset ((k \ \mathsf{says} \ s) \ @ \ [u_1, u_2])$

The @ connective has a trivial interaction with itself – nested @ connectives can be reduced to the innermost only.

12. $\vdash ((s \ @ \ [u_1, u_2]) \ @ \ [u_1', u_2']) \equiv (s \ @ \ [u_1, u_2])$

Finally, axiom (S) of BL_S from §3.1.1 can be generalized in BL by internalizing the side condition $\Sigma \vdash k \succeq k'$ as a constraint.

13. $\vdash (k' \succeq k) \supset ((k' \ \mathsf{says} \ s) \supset (k \ \mathsf{says} \ s))$

Interaction of constraints with says and @. There are two interactions between constraints and says, and constraints and @ that deserve careful scrutiny. First, $c \supset (k \ \mathsf{says} \ c)$ for every k and c, which means that every true constraint is supported by every principal. This supports the idea that there is a unique definition of constraint satisfaction on which all principals agree. This may not be case for other formulas since, in general, it is not the case that $s \supset (k \ \mathsf{says} \ s)$. Further, $(k \ \mathsf{says} \ c)$ does not imply c. This prevents principals from changing the universal meaning of constraints simply by asserting new constraints.

Second, it is the case that $(c \ @ \ [u_1, u_2]) \equiv (c \ @ \ [u_1', u_2'])$. This a consequence of the fact that the truth of a constraint is independent of time: either c holds on all intervals or it holds on none. All these interactions are summarized below.

14. $\vdash c \supset (k \ \mathsf{says} \ c)$

15. $\not\vdash (k \text{ says } c) \supset c$

16. $\vdash (c \mathbin{@} [u_1, u_2]) \equiv (c \mathbin{@} [u_1', u_2'])$

Interaction of interpreted atoms with other connectives. The interaction of interpreted atoms with most connectives of BL is unremarkable; they behave almost like their uninterpreted counterparts. However, interpreted atoms interact with says and @ in a manner that is similar to that described previously for constraints. In particular, the following properties hold in BL.

17. $\vdash i \supset (k \text{ says } i)$

18. $\not\vdash (k \text{ says } i) \supset i$

19. $\vdash (i \mathbin{@} [u_1, u_2]) \equiv (i \mathbin{@} [u_1', u_2'])$

Property (17) implies (as for constraints) that all principals agree on a single definition of interpreted predicates, which should be intuitive since there is a single system state at the point of policy enforcement. Property (18) prevents principals from changing this unique interpretation through assertions.

Property (19) is counter-intuitive, and reflects an important design choice not only in BL but also in PCFS. The apparent problem with the property is that, in practice, the truth of interpreted atoms *does* change with time as the system state changes, so $i \mathbin{@} [u_1, u_2]$ should not imply $i \mathbin{@} [u_1', u_2']$ for arbitrary intervals $[u_1, u_2]$ and $[u_1', u_2']$. The reason that this implication holds in BL is that interpreted atoms are evaluated in a *fixed* system state that is explicitly assumed, and is supposed to represent the state prevailing at the time of access. This state is denoted by the symbol E in §4.2. Consequently, the statement "i is true" implicitly means that "i is true in the explicitly assumed state E", and is, therefore, independent of time.

The ramification of this design choice is that *history* of system state cannot be captured by interpreted atoms and explicit time in an intuitive manner. While this may seem limiting, it is necessitated by practical concerns: if $i \mathbin{@} [u_1, u_2]$ did indeed mean that i were true during the interval $[u_1, u_2]$, then any proof verifier would need a record of the entire history (and possibly future) of system state in order to check proofs. This is clearly impractical. On the other hand, limiting system state to one point in time does not really reduce expressiveness: if some access control policy were to rely on a predicate over system state having been true in the past, this can still be represented in BL by requiring that there be explicit evidence – either an element of system state or a certificate – *still valid at the time of access* that witnesses this fact. Clearly, requiring such persistent evidence is no harder than requiring the reference monitor to maintain a record of the entire history, and is in fact, a better design choice since it requires the policy to make explicit what evidence from the past is necessary to verify proofs.

4.1.2 Expressible Policy Idioms

Using explicit time, constraints, and interpreted predicates, several new policy idioms in addition those already expressible in BL_S (§3.1.2) become expressible in BL. This section lists some of these idioms.

Certificate expiration. The simplest use of explicit time is to accurately represent expiration of certificates in the logic. For example, if Alice signs a certificate allowing Bob read access to file secret.txt from February 01, 2009 to February 28, 2009, this can be represented in BL as the formula (Alice says (may Bob secret.txt read)) @ [2009:02:01, 2009:02:28].[2] The interaction of the @ connective with constraints in BL ensures that this time interval is respected during enforcement.

Anachronistic references. Explicit time can also be used to represent policies that depend on facts having been true at explicit time points in the past. This often happens in policies that represent a change of scenario. Suppose, for instance, that a university UV allows its alumni to continue to access their files for six months after they leave UV. This policy can be expressed using a combination of explicit time and constraints in BL. Let the predicate `alumni` k T mean that k became an alumni at time T, let the constraint `is` T T' mean that T and T' are equal, and let $180d$ denote a time period of six months. Then, UV may represent this policy as follows.

$$\begin{aligned}&\mathsf{UV}\ \mathsf{says}\ \forall k, f, T, T'.\\ &\quad (((\texttt{alumni}\ k\ T) \wedge ((\texttt{mayaccess}\ k\ f)\ @\ [T, T]) \wedge (\texttt{is}\ T'\ (T + 180d)))\\ &\qquad \supset ((\texttt{mayaccess}\ k\ f)\ @\ [T, T']))\end{aligned}$$

This policy rule states that if k became an alumni at time T and k could access file f at time T, then k may access file f during the interval $[T, T + 180d]$ as well. The constraint `is` T T' and the arithmetic operator $+$ used in this policy rule are supported in the implementation of BL in PCFS (see §4.3).

State dependent policies. Access is sometimes dependent on the state of the system, which is itself not modeled in the policy as a certificate. This can happen when access rights change during different stages of a workflow. Such policies can be expressed in BL using interpreted predicates. For example, in PCFS, files go through two states: default and governed. A newly created file is in the default state, and in this state the owner of the file has all access to the file, whereas in the governed state other applicable policy rules determine access to the file. Let the interpreted predicates **state** f S and **owner** f k respectively mean that file f is in state S, and that the owner of file f is principal k. Then,

[2] In Section 4.3 we describe the representation of digital certificates using basic judgments of BL instead of formulas. Although logically equivalent, that representation is slightly simpler and closer to the actual implementation of BL in PCFS.

the access policy for the default state may be expressed using the following formula.[3]

$$\mathsf{admin\ says}\ \forall k, f.\ (((\mathbf{state}\ f\ \mathsf{default}) \wedge (\mathbf{owner}\ f\ k)) \supset (\mathtt{mayaccess}\ k\ f))$$

Other examples of policies that depend on state, and in particular on attributes of files may be found in §4.3.3 and §8.

Thresholding. It was mentioned in §3.7 that it is difficult to express without constraints policies that allow access only when m out of n designated principals approve. However, with constraints, such thresholding is easy to express. For example, the following policy rule states that Alice supports s if at least three good principals also support s. The constraint `different` k_1 k_2 k_3 means that the three principals k_1, k_2, and k_3 are distinct. (This example is based on a similar example in a paper on the policy language SecPAL [23].)

$$\begin{array}{l}\mathsf{Alice\ says}\ \forall k_1, k_2, k_3. \\ \quad (((k_1\ \mathsf{says}\ s) \wedge (k_2\ \mathsf{says}\ s) \wedge (k_3\ \mathsf{says}\ s) \wedge (\mathtt{good}\ k_1) \wedge (\mathtt{good}\ k_2) \wedge (\mathtt{good}\ k_3) \wedge \\ \quad\ (\mathtt{different}\ k_1\ k_2\ k_3)) \supset s)\end{array}$$

4.2 Structural Proof Theory

We now turn to the centerpiece of this thesis – the proof theory of BL. We describe a natural deduction system and a sequent calculus for the logic, and study their metatheoretic properties. The technical content in this section generalizes structural proof theory for BL_S (§3.2). Before presenting the inference systems we discuss how constraint domains and interpreted predicates are formally represented in the logic since they are crucial to both natural deduction and the sequent calculus.

4.2.1 Constraints and Interpreted Predicates

Unlike uninterpreted predicates which are established by applying inference rules of the logic to hypotheses, the rules for establishing constraints and interpreted predicates are not stipulated in BL. Instead, both constraints and interpreted predicates are established through external solvers, which we formally reflect in the logic via judgments without any specific rules.

Representation of constraint domains. Let $\Psi = c_1, \ldots, c_n$ denote a set of constraints, possibly containing free variables that are implicitly assumed to be universally quantified. Let c be another constraint and let Σ be a sorting whose domain contains all the free variables of Ψ and c. We write $\Sigma; \Psi \models c$ if and only if for every grounding substitution θ whose domain includes the domain of Σ, it is the case that $c_1\theta, \ldots, c_n\theta$ *entail* $c\theta$. This

[3] This policy rule is merely an illustration. In the actual implementation of PCFS, the state of a file is represented through an extended attribute which can be represented in BL through a generic interpreted predicate **has_xattr**, and the owner is given access in the default state by procaps, not a policy rule. See §7 for details.

entailment may be classical, i.e. the constraint solver may simply check that $\neg c_1\theta \vee \ldots \vee \neg c_n\theta \vee c\theta$ holds.

Clearly, the constraint domain must support universally quantified variables. For certain fragments of the logic, this requirement can be waived. In particular, if both universal quantification and implication are disallowed on the right hand side of hypothetical judgments and existential quantification is disallowed on the left, then the constraint domain does not need to take into account universally quantified variables. This is because only the rules ($\supset$R), ($\forall$R), and ($\exists$L) of the sequent calculus (§4.2.4) introduce variables in Σ. Although this restricted fragment is quite expressive, the implementation of the solver for constraints in PCFS takes into account universally quantified variables and does not need this restriction.

Proof theory of BL uses the judgment $\Sigma; \Psi \models c$ as a "black-box", and is, therefore, oblivious to the details of the constraint solver used. However, in order to obtain metatheoretic properties of the inference systems, we make the following assumptions about the constraint domain.

(C-hyp) $\Sigma; \Psi, c \models c$.

(C-weaken) $\Sigma; \Psi \models c$ implies both $\Sigma; \Psi, c' \models c$ and $\Sigma, x{:}\sigma; \Psi \models c$.

(C-cut) $\Sigma; \Psi \models c$ and $\Sigma; \Psi, c \models c'$ imply $\Sigma; \Psi \models c'$.

(C-subst) $\Sigma, x{:}\sigma; \Psi \models c$ and $\Sigma \vdash t : \sigma$ imply $\Sigma; \Psi[t/x] \models c[t/x]$.

(C-refl-time) $\Sigma; \Psi \models u \leq u$.

(C-trans-time) $\Sigma; \Psi \models u \leq u'$ and $\Sigma; \Psi \models u' \leq u''$ imply $\Sigma; \Psi \models u \leq u''$.

(C-refl-prin) $\Sigma; \Psi \models k \succeq k$.

(C-trans-prin) $\Sigma; \Psi \models k \succeq k'$ and $\Sigma; \Psi \models k' \succeq k''$ imply $\Sigma; \Psi \models k \succeq k''$.

(C-inf-time) $\Sigma; \Psi \models u \leq +\infty$ and $\Sigma; \Psi \models -\infty \leq u$

(C-loc-prin) $\Sigma; \Psi \models \ell \succeq k$

(C-hyp), (C-weaken), and (C-cut) should hold for any reasonable constraint domain, simply by definition of entailment. (C-subst) means that the constraint domain accounts for universally quantified variables correctly. (C-refl-time) and (C-trans-time) mean that the constraint domain must treat the relation $u \leq u'$ as a preorder. (C-refl-prin) and (C-trans-prin) impose a similar condition on $k \succeq k'$. (C-inf-time) and (C-loc-prin) ensure that $+\infty$ and $-\infty$ are treated as the greatest and the least time points respectively, and that ℓ is the strongest principal.

How easy is it to implement a decision procedure for solving $u \leq u'$ and $k \succeq k'$, the two forms of constraints mandated by BL? It turns out that this is extremely easy. In each case, we only need to take a reflexive transitive closure of the relations assumed in Ψ and check that the goal c lies in the result. Both the front end (proof search, proof verifier) and the back end (reference monitor) of PCFS implement these decision procedures. A typical check takes around 2μs on a 2.4GHz Intel Core 2 Duo processor.

Representation of the solver for interpreted predicates. Interpreted predicates are checked directly on the prevailing state of the system. In the logic, the state of the system is abstractly represented as a set of interpreted atoms, denoted E. The judgment $\Sigma; E \models i$ means that for all grounding substitutions θ whose domain contains the domain of Σ, $i\theta \in E\theta$. In order to prove metatheoretic properties of BL's inference systems, we make the following assumptions about this judgment, all of which should be intuitive from its definition.

(S-hyp) $\Sigma; E, i \models i$.

(S-weaken) $\Sigma; E \models i$ implies both $\Sigma; E, E' \models i$ and $\Sigma, x{:}\sigma; \Psi \models c$.

(S-cut) $\Sigma; E \models i$ and $\Sigma; E, i \models i'$ imply $\Sigma; E \models i'$.

(S-subst) $\Sigma, x{:}\sigma; E \models i$ and $\Sigma \vdash t : \sigma$ imply $\Sigma; E[t/x] \models i[t/x]$.

In an actual implementation it may be infeasible to represent the entire system state explicitly in E because it may be very large or even infinite. Accordingly, in the implementation of PCFS, only certain atoms in E are represented explicitly. These are atoms that are added to E in a proof rule, e.g., (interE) in the natural deduction system (§4.2.2). The rest of the state is left implicit in the system, and is checked directly by the solver.

4.2.2 Natural Deduction

Our presentation of BL's proof theory, as also for the case of BL_S, is based in Martin-Löf's judgmental description of type theory [99] and draws on its refinements in the work of Pfenning and Davies [115]. More directly, the treatment of says in BL is based on that in BL_S, and the treatment of time is based on that in η logic [53, 54]. This section describes natural deduction for BL whereas §4.2.4 covers the sequent calculus.

Basic judgments in BL are relativized to time; absolute truth of formulas independent of time cannot be asserted in BL. We use two basic judgments (denoted J) in our presentation: $s \circ [u_1, u_2]$ which means that s holds throughout the closed interval $[u_1, u_2]$, and k claims $s \circ [u_1, u_2]$ which means that k supports or claims throughout the interval $[u_1, u_2]$ that s is true. The symbol $\circ$ is read "on" or "throughout". The two basic judgments do not entail each other in general. $s \circ [u_1, u_2]$ is *internalized* in the syntax of formulas as $s \mathbin{@} [u_1, u_2]$ whereas k claims $s \circ [u_1, u_2]$ is internalized through a combination of two connectives as $(k \text{ says } s) \mathbin{@} [u_1, u_2]$.

Hypothetical Judgments. Hypothetical judgments of BL, which are the subjects of its inference rules, take the form $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$. Ψ and E are assumed constraints and interpreted atoms respectively. The hypotheses Γ is a multiset of basic judgments. ν is called the view of the hypothetical judgment. It is a triple that contains a principal and two time points, written k_0, u_b, u_e. The presence of time points u_b, u_e in views is motivated by practical considerations of representing policies intuitively, and is justified in §4.4.

Basic Judgments	J	::=	$s \circ [u_1, u_2] \mid k \ \mathsf{claims}\ s \circ [u_1, u_2]$
Hypothetical Constraints	Ψ	::=	$c_1 \dots c_n$
System State	E	::=	$i_1 \dots i_n$
Views	ν	::=	k_0, u_b, u_e
Hypotheses	Γ	::=	$J_1 \dots J_n \qquad (n \geq 0)$
Hypothetical Judgments			$\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$

Analogous to inference in BL_S, natural deduction for BL is guided by several principles that relate its judgments. As for BL_S, the view principle explains the role of views in inference and is incorporated as a rule in natural deduction whereas two other principles, the substitution principle and the claim principle, are proved to be admissible (§4.2.3). In addition, there is a fourth principle in BL, called the *time subsumption principle*. We describe these four principles below, starting with time subsumption.

Time subsumption principle. $s \circ [u_1', u_2']$ entails $s \circ [u_1, u_2]$ if $u_1' \leq u_1$ and $u_2 \leq u_2'$.

The time subsumption principle means that if s is known to be true throughout an interval $[u_1, u_2]$, it must also be true on every subinterval $[u_1', u_2']$. This principle is incorporated into the following hypothesis rule of natural deduction. For conclusions of hypothetical judgments, we prove the principle as a theorem (Theorem 4.4).

$$\frac{\Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\mathrm{hyp}$$

The time subsumption principle is very important for the implementation of PCFS. This is explained in §4.3.

View principle. While reasoning in view k_0, u_b, u_e, the assumption $k \ \mathsf{claims}\ s \circ [u_1, u_2]$ entails $s \circ [u_1, u_2]$ if $k \succeq k_0$, $u_1 \leq u_b$ and $u_e \leq u_2$.

Together with the time subsumption principle, the view principle results in the following rule in natural deduction:

$$\frac{\begin{array}{ccc}\nu = k, u_b, u_e & \Sigma; \Psi \models u_1' \leq u_1 & \Sigma; \Psi \models u_2 \leq u_2' \\ \Sigma; \Psi \models u_1' \leq u_b & \Sigma; \Psi \models u_e \leq u_2' & \Sigma; \Psi \models k' \succeq k\end{array}}{\Sigma; \Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\mathrm{claims}$$

Substitution principle. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']$ imply $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']$.

The substitution principle means that if $s \circ [u_1, u_2]$ is assumed explicitly in a proof of $s' \circ [u_1', u_2']$ and the former can be proved directly, then so can the latter. We prove later that this principle is admissible in BL (Theorem 4.5).

$$\frac{\Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\text{hyp}$$

$$\frac{\begin{array}{c}\nu = k, u_b, u_e \qquad \Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2' \\ \Sigma; \Psi \models u_1' \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2' \qquad \Sigma; \Psi \models k' \succeq k\end{array}}{\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\text{claims}$$

$$\frac{\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysI}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma, k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}\mathsf{saysE}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} (s \mathbin{@} [u_1, u_2]) \circ [u_1', u_2']}@\text{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \mathbin{@} [u_1, u_2] \circ [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1'', u_2'']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1'', u_2'']}@\text{E}$$

$$\frac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} c \circ [u_1, u_2]}\text{consI} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} c \circ [u_1, u_2] \qquad \Sigma; \Psi, c; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}\text{consE}$$

$$\frac{\Sigma; E \models i}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} i \circ [u_1, u_2]}\text{interI} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} i \circ [u_1, u_2] \qquad \Sigma; \Psi; E, i; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}\text{interE}$$

Figure 4.2: BL: Natural deduction, part 1

Claim principle. $\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Gamma, k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']$ imply $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']$.

The claim principle defines the meaning of the judgment $k \ \mathsf{claims}\ s \circ [u_1, u_2]$. According to the principle, the judgment $k \ \mathsf{claims}\ s \circ [u_1, u_2]$ can be substituted by a proof of $s \circ [u_1, u_2]$ provided that the latter was obtained in the context k, u_1, u_2, and only from claims of principals. The restriction operator $\Gamma|$ removes from Γ all judgments of the form $r \circ [u_b, u_e]$:

$$\Gamma| = \{(k_0 \ \mathsf{claims}\ r \circ [u_b, u_e]) \in \Gamma\}$$

We prove the admissibility of the claim principle as a theorem (Theorem 4.6).

Inference rules. Figures 4.2 and 4.3 show the rules of the natural deduction proof system. Figure 4.2 contains rules pertaining to hypotheses and connectives other than those of first-order logic. Figure 4.3 contains rules pertaining to connectives of first-order logic. As usual, we have introduction and elimination rules for each connective (marked I and E respectively). For a syntactic entity Ξ, $\Xi[t/x]$ denotes the capture avoiding substitution of term t for variable x in Ξ.

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]} \wedge \mathrm{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2]} \wedge \mathrm{E}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2]} \wedge \mathrm{E}_2$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2]} \vee \mathrm{I}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2]} \vee \mathrm{I}_2$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2] \\ \Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma, s_2 \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']} \vee \mathrm{E}$$

$$\frac{}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \top \circ [u_1, u_2]} \top \mathrm{I} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \bot \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1', u_2']} \bot \mathrm{E}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \supset \mathrm{I}$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \\ \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1', u_2'] \qquad \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1', u_2']} \supset \mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \forall x{:}\sigma.s \circ [u_1, u_2]} \forall \mathrm{I} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \forall x{:}\sigma.s \circ [u_1, u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s[t/x] \circ [u_1, u_2]} \forall \mathrm{E}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s[t/x] \circ [u_1, u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \exists x{:}\sigma.s \circ [u_1, u_2]} \exists \mathrm{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \exists x{:}\sigma.s \circ [u_1, u_2] \qquad \Sigma, x{:}\sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']} \exists \mathrm{E}$$

Figure 4.3: BL: Natural deduction, part 2

Rules (hyp) and (claims) in Figure 4.2 allow the use of hypotheses of the forms $s \circ [u_1, u_2]$ and k claims $s \circ [u_1, u_2]$ respectively. As mentioned earlier, they capture the time subsumption and view principles in the proof system. Rule (saysI) can be justified using the claim principle. Since k says $s \circ [u_1, u_2]$ is logically equivalent to k claims $s \circ [u_1, u_2]$, the claim principle implies that $(k$ says $s) \circ [u_1, u_2]$ should be provable whenever $s \circ [u_1, u_2]$ can be proved in the view k, u_1, u_2 with hypotheses $\Gamma|$. The latter is exactly the premise of (saysI). It should be observed that (saysI) is the only rule in natural deduction that changes the view, and further that there is a strong interaction between says and explicit time in BL – the two time points in a view are obtained from the last application of (saysI), progressing

backwards on a derivation. The elimination rule (saysE) is straightforward; it means that a proof of k says $s \circ [u_1, u_2]$ can be used to substitute the equivalent judgmental form k claims $s \circ [u_1, u_2]$ from another proof with the same hypotheses.

Rule (@I) states that in order to establish s @ $[u_1, u_2]$ during any interval $[u_1', u_2']$, it suffices to establish $s \circ [u_1, u_2]$. Dually, rule (@E) means that s @ $[u_1, u_2] \circ [u_1', u_2']$ is stronger than $s \circ [u_1, u_2]$. Together the rules imply that s @ $[u_1, u_2] \circ [u_1', u_2']$ and $s \circ [u_1, u_2]$ are equivalent as judgments, as well as property (12) from §4.1.1.

According to rule (consI), $c \circ [u_1, u_2]$ may be established by showing c in the prevailing constraint hypotheses Ψ. Dually, if $c \circ [u_1, u_2]$ can be established then the constraint c may be assumed in Ψ (rule (consE)). The two rules together mean that the interpretation of constraints is independent of time, which was also discussed earlier in §4.1.1. Rules (interI) and (interE) for interpreted atoms are very similar to rules (consI) and (consE) respectively. Indeed, constraints and interpreted predicates are so similar to each other from a proof-theoretic perspective that they can be merged into one syntactic class without affecting the proof theory and metatheory of BL significantly. However, since the truth of interpreted predicates changes with system state, whereas that of constraints does not, the two must be treated differently during proof verification in PCFS (§5). Hence we maintain a distinction between their syntactic classes.

Since most connectives of first-order logic commute with @ in BL (see §4.1.1), in their corresponding inference rules in Figure 4.3, time intervals do not change. For example, to establish $s_1 \wedge s_2 \circ [u_1, u_2]$, it suffices to establish $s_1 \circ [u_1, u_2]$ and $s_2 \circ [u_1, u_2]$ (rule ($\wedge$I)). Implication is the only connective of first-order logic that has an interesting interaction with explicit time. As mentioned in §4.1.1, having a proof of $(s_1 \supset s_2)$ @ $[u_1, u_2]$ is equivalent to having a proof of s_2 @ $[x_1, x_2]$ from the assumption s_1 @ $[x_1, x_2]$ for every subinterval $[x_1, x_2]$ of $[u_1, u_2]$. The rule ($\supset$I) lifts this intuition to judgments: in order to establish $s_1 \supset s_2 \circ [u_1, u_2]$, it suffices to show that for any two time variables x_1, x_2 such that $u_1 \leq x_1$ and $x_2 \leq u_2$, it is the case that $s_1 \circ [x_1, x_2]$ entails $s_2 \circ [x_1, x_2]$. Dually, the rule ($\supset$E) means that if there are proofs of $s_1 \supset s_2 \circ [u_1, u_2]$ and $s_1 \circ [u_1', u_2']$, where $[u_1', u_2'] \subseteq [u_1, u_2]$, then there is also a proof of $s_2 \circ [u_1', u_2']$.

A note on analysis of constraints. As should be evident from the rules of Figures 4.2 and 4.3, it is impossible to analyze the structure of constraints in proofs of BL. In particular, BL lacks two common rules that previous descriptions of constraint domains in logic have allowed (see, e.g., [84]). The first of these rules allows a deduction of any formula from a contradictory constraint. For example, it makes $(1 \leq 0) \supset s$ admissible. The second rule allows a case analysis on constraints. Were the second rule to be admitted in BL, it would suffice to show that s holds on $[u_1, u]$ and also on $[u, u_2]$, possibly through two different proofs, in order to conclude that s holds on $[u_1, u_2]$. Thus $((s$ @ $[u_1, u]) \wedge (s$ @ $[u, u_2])) \supset (s$ @ $[u_1, u_2])$ would be provable. In BL, we refrain from allowing any such analysis of constraints within the logic for three reasons. First, because constraints are not justified through explicit evidence, allowing their analysis through the logic's rules may result in reduced accountability in proofs. Second, this design decision allows us to prove that any proof term which witnesses s @ $[u_1, u_2]$ also witnesses, without change,

s @ $[u_1', u_2']$ for any $[u_1', u_2'] \subseteq [u_1, u_2]$ (Theorem 5.5). This result is of practical importance in the implementation of PCFS, as explained in §4.3.2. Third, it is not clear whether an automatic theorem prover can decide when to analyze constraints during proof search, so describing a complete proof search strategy for the logic may be impossible if analysis of constraints is allowed.

4.2.3 Metatheory of Natural Deduction

We prove several metatheoretic properties of the natural deduction system of BL, many of which generalize properties of BL_S (§3.2.2). Besides structural properties (weakening and contraction), we show that instantiation as well as the substitution, claim, and time subsumption principles are admissible.

Theorem 4.1 (Weakening and Contraction)**.** *The following hold:*

1. *(Weakening)*
   - *(a)* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *implies* $\Sigma, x{:}\sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$.
   - *(b)* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi, c; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$.
   - *(c)* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E, i; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$.
   - *(d)* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma, J \vdash^{\nu} s \circ [u_1, u_2]$.
2. *(Contraction)* $\Sigma; \Psi; E; \Gamma, J, J \vdash^{\nu} s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma, J \vdash^{\nu} s \circ [u_1, u_2]$.

Further the derivation in the consequent of each statement has a depth no more than that of the antecedent.[4]

Proof. By separate induction on the given derivation for each property. □

Theorem 4.2 (Instantiation)**.** $\Sigma, x{:}\sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *and* $\Sigma \vdash t : \sigma$ *imply* $\Sigma; \Psi[t/x]; E[t/x]; \Gamma[t/x] \vdash^{\nu[t/x]} s[t/x] \circ [u_1[t/x], u_2[t/x]]$

Proof. By induction on the derivation of $\Sigma, x{:}\sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$. □

The following subsumption property for views is analogous to Theorem 3.4 for BL_S.

Theorem 4.3 (View subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$
2. $\nu = k_0, u_b, u_e$
3. $\Sigma; \Psi \models k_0 \succeq k_0'$, $\Sigma; \Psi \models u_b \leq u_b'$, *and* $\Sigma; \Psi \models u_e' \leq u_e$.

[4] The depth of a derivation is defined as the maximum number of BL's inference rules on a path in the derivation that starts from its conclusion and ends at a leaf. Rules needed to establish auxiliary judgments like $\Sigma \vdash t : \sigma$, $\Sigma; \Psi \models c$, and $\Sigma; E \models i$ are not part of BL's inference rules and do not count towards the depth.

4. $\nu' = k'_0, u'_b, u'_e$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu'} s \circ [u_1, u_2]$ *by a derivation of smaller or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. There is only one interesting case which is shown below.

Case. $$\dfrac{\begin{array}{ccc}\nu = k_0, u_b, u_e & \Sigma; \Psi \models u'_1 \leq u_1 & \Sigma; \Psi \models u_2 \leq u'_2 \\ \Sigma; \Psi \models u'_1 \leq u_b & \Sigma; \Psi \models u_e \leq u'_2 & \Sigma; \Psi \models k' \succeq k_0\end{array}}{\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u'_1, u'_2] \vdash^{\nu} s \circ [u_1, u_2]}\text{claims}$$

To show: $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u'_1, u'_2] \vdash^{\nu'} s \circ [u_1, u_2]$

1. $\Sigma; \Psi \models u_b \leq u'_b$ (Assumption 3)
2. $\Sigma; \Psi \models u'_1 \leq u_b$ (Premise)
3. $\Sigma; \Psi \models u'_1 \leq u'_b$ ((C-trans-time) from §4.2.1 on 1,2)
4. $\Sigma; \Psi \models u'_e \leq u_e$ (Assumption 3)
5. $\Sigma; \Psi \models u_e \leq u'_2$ (Premise)
6. $\Sigma; \Psi \models u'_e \leq u'_2$ ((C-trans-time) from §4.2.1 on 4,5)
7. $\Sigma; \Psi \models k_0 \succeq k'_0$ (Assumption 3)
8. $\Sigma; \Psi \models k' \succeq k_0$ (Premise)
9. $\Sigma; \Psi \models k' \succeq k'_0$ ((C-trans-prin) from §4.2.1 on 7,8)
10. $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u'_1, u'_2] \vdash^{\nu'} s \circ [u_1, u_2]$

(Rule (claims) on 2nd,3rd premises and 3,6,9)

The depths of the given derivation and the derivation constructed above are each equal to 1. □

Next we show that the time subsumption principle, substitution principle, and the claim principle are admissible in BL.

Theorem 4.4 (Time subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$
3. $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m]$.

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. The proof appeals to Theorem 4.3 and a lemma about substitution of constraints. See Theorem B.2 in Appendix B for details of the lemma as well as some of the interesting cases of the proof. □

Like many other theorems in this section, ensuring that the time subsumption principle holds requires care. For example, if we were to replace the rule (@E) by the following rule (@E'), which is also admissible in BL and perhaps a more obvious choice, then the time subsumption principle would no longer hold.

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \mathbin{@} [u_1, u_2] \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]} \text{@E'}$$

Theorem 4.5 (Substitution)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2']$ and a case analysis of its last rule. The only interesting case of the proof, which appeals to Theorem 4.4 is shown below.

Case. $\dfrac{\Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2}{\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1', u_2']}$ hyp

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ (Assumption)
2. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$ (Premises)
3. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1', u_2']$ (Theorem 4.4 on 1,2)

□

Theorem 4.6 (Claim)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]$
2. $\Sigma; \Psi; E; \Gamma, k \mathsf{\ claims\ } s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2']$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u'_1, u'_2]$, and case analysis of the last rule in it. The interesting cases are shown below.

Case. $$\frac{\begin{array}{ccc}\nu = k', u_b, u_e & \Sigma; \Psi \models u_1 \leq u'_1 & \Sigma; \Psi \models u'_2 \leq u_2 \\ \Sigma; \Psi \models u_1 \leq u_b & \Sigma; \Psi \models u_e \leq u_2 & \Sigma; \Psi \models k \succeq k'\end{array}}{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \vdash^{\nu} s \circ [u'_1, u'_2]}\text{claims}$$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u'_1, u'_2]$

1. $\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]$ (Assumption 1)
2. $\Sigma; \Psi; E; \Gamma \vdash^{k, u_1, u_2} s \circ [u_1, u_2]$ (Weakening Theorem 4.1 on 1)
3. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ (Theorem 4.3 on 2 and premises 4–6)
4. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u'_1, u'_2]$ (Theorem 4.4 on 3 and premises 2,3)

Case. $$\frac{\Sigma; \Psi; E; \Gamma|, k \text{ claims } s \circ [u_1, u_2] \vdash^{k', u'_1, u'_2} s' \circ [u'_1, u'_2]}{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \vdash^{\nu} k' \text{ says } s' \circ [u'_1, u'_2]}\text{saysI}$$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} (k' \text{ says } s') \circ [u'_1, u'_2]$

1. $(\Gamma|)| = \Gamma|$ (Definition)
2. $\Sigma; \Psi; E; (\Gamma|)| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]$ (1 and Assumption 1)
3. $\Sigma; \Psi; E; \Gamma| \vdash^{k', u'_1, u'_2} s' \circ [u'_1, u'_2]$ (i.h. on 2 and premise)
4. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} (k' \text{ says } s') \circ [u'_1, u'_2]$ (Rule (saysI) on 3)

□

4.2.4 Sequent Calculus

As discussed in §3.2.3, in a sequent calculus inference rules apply to both the hypotheses and conclusion of hypothetical judgments, and always decompose connectives when going from the conclusion to premises. Hypothetical judgments in a sequent calculus are called sequents. They have the same form as the hypothetical judgments in natural deduction, but we use a different entailment symbol $\xrightarrow{\nu}$ in sequents to distinguish the two inference systems.

$$\text{Sequents} \;::=\; \Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1, u_2]$$

Inference rules in a sequent calculus are categorized as either left or right, marked L and R respectively, according to the location of the formula they decompose relative to the entailment symbol. The rules of the sequent calculus for BL are shown in Figures 4.4 and 4.5. Rules pertaining to the use of hypotheses and those pertaining to connectives not in first-order logic are in Figure 4.4, while the remaining rules are in Figure 4.5.

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

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

$$\frac{\Sigma;\Psi;E;\Gamma| \xrightarrow{k,u_1,u_2} s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1,u_2]}\mathsf{saysR}$$

$$\frac{\Sigma;\Psi;E;\Gamma, k\ \mathsf{says}\ s \circ [u_1,u_2], k\ \mathsf{claims}\ s \circ [u_1,u_2] \xrightarrow{\nu} r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Gamma, k\ \mathsf{says}\ s \circ [u_1,u_2] \xrightarrow{\nu} r \circ [u_1',u_2']}\mathsf{saysL}$$

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

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

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

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} i \circ [u_1,u_2]}\text{interR} \qquad \frac{\Sigma;\Psi;E, i;\Gamma, i \circ [u_1,u_2] \xrightarrow{\nu} r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Gamma, i \circ [u_1,u_2] \xrightarrow{\nu} r \circ [u_1',u_2']}\text{interL}$$

Figure 4.4: BL: Sequent calculus, part 1

Rule (init) in Figure 4.4 allows an atomic hypotheses $p \circ [u_1', u_2']$ to be used to conclude $p \circ [u_1, u_2]$ if $[u_1', u_2']$ is a superset of $[u_1, u_2]$. The generalization of this rule to arbitrary formulas corresponds exactly to the rule (hyp) from natural deduction, and is proved to be admissible (Theorem 4.13). Rule (claims) captures exactly the view principle from §4.2.2 in the sequent calculus. It should be noted that its homonym in natural deduction is stronger since that also incorporates the time subsumption principle. However, the two rules are equivalent in the presence of the time subsumption principle (Theorem 4.14).

All other rules decompose connectives. The right rules for each connective are identical to corresponding introduction rules in natural deduction, with the exception of the difference in the entailment symbol. Left rules in the sequent calculus fulfill the same purpose as elimination rules in natural deduction. However, they decompose connectives in the hypotheses, when the rule is read from conclusion to premises. Rule (saysL) in Figure 4.4 means that if k says $s \circ [u_1, u_2]$ is assumed in a proof, then so may the hypothesis k claims $s \circ [u_1, u_2]$, since the two are logically equivalent to each other. Rule (consL)

$$\frac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_1, u_2]} \wedge\mathrm{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma, s_1 \wedge s_2 \circ [u_1, u_2], s_1 \circ [u_1, u_2], s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, s_1 \wedge s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']} \wedge\mathrm{L}$$

$$\frac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \vee s_2 \circ [u_1, u_2]} \vee\mathrm{R}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \vee s_2 \circ [u_1, u_2]} \vee\mathrm{R}_2$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, s_1 \vee s_2 \circ [u_1, u_2], s_1 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Gamma, s_1 \vee s_2 \circ [u_1, u_2], s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']\end{array}}{\Sigma; \Psi; E; \Gamma, s_1 \vee s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']} \vee\mathrm{L}$$

$$\frac{}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \top \circ [u_1, u_2]} \top\mathrm{R} \qquad \frac{}{\Sigma; \Psi; E; \Gamma, \bot \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']} \bot\mathrm{L}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \supset\mathrm{R}$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} s_1 \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2], s_2 \circ [u_1', u_2'] \xrightarrow{\nu} r \circ [u_1'', u_2''] \qquad \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1'', u_2'']} \supset\mathrm{L}$$

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \forall x{:}\sigma.s \circ [u_1, u_2]} \forall\mathrm{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.s \circ [u_1, u_2], s[t/x] \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2'] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']} \forall\mathrm{L}$$

$$\frac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s[t/x] \circ [u_1, u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \exists x{:}\sigma.s \circ [u_1, u_2]} \exists\mathrm{R}$$

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma, \exists x{:}\sigma.s \circ [u_1, u_2], s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, \exists x{:}\sigma.s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']} \exists\mathrm{L}$$

Figure 4.5: BL: Sequent calculus, part 2

means that the assumption $c \circ [u_1, u_2]$ entails the constraint c, which we already justified in §4.1.1. Rule (interL) is similar, except that it applies to interpreted atoms. The left rules in Figure 4.5 correspond to the properties in §4.1.1. The only remarkable rule here is ($\supset$L), which allows the assumption $s_2 \circ [u_1', u_2']$ to be introduced (second premise) if $s_1 \supset s_2 \circ [u_1, u_2]$ holds during some interval $[u_1, u_2] \supseteq [u_1', u_2']$, and $s_1 \circ [u_1', u_2']$ is provable

(first premise). This follows from our intuitive understanding of implication in BL.

Example 4.7 (Properties from §4.1.1)**.** Although all judgments in BL are relativized to time, we may *define* a priori provability of formula s, written $\vdash s$, as an abbreviation for $\Sigma;\cdot;\cdot;\cdot \xrightarrow{\nu} s \circ [-\infty,+\infty]$ where Σ assigns sorts to all variables in s and ν is a view made of three fresh constants. With this definition, all properties of §4.1.1, including those of the form $\nvdash s$, can be established using the rules in Figures 4.4 and 4.5. In addition, $\nvdash \bot$, so BL is consistent.

4.2.5 Metatheory of the Sequent Calculus

Next we prove several important metatheorems for the sequent calculus of BL including admissibility of cut, which encompasses both the substitution principle and the claim principle (Theorems 4.5 and 4.6), as well as the identity principle, which generalizes the (init) rule of Figure 4.4 from uninterpreted atoms to arbitrary formulas. These two theorems together are often considered proof-theoretic analogues of soundness and completeness for inference systems, particularly because they imply, and are essential in the proof of, equivalence of natural deduction and the sequent calculus (Theorem 4.14). Admissibility of cut also implies that natural deduction proofs can be reduced to a normal form, a fact that we prove in §4.5.

Theorem 4.8 (Weakening and Contraction)**.** *The following hold:*

1. *(Weakening)*
   - *(a)* $\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$ *implies* $\Sigma,x{:}\sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$.
   - *(b)* $\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$ *implies* $\Sigma;\Psi,c;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$.
   - *(c)* $\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$ *implies* $\Sigma;\Psi;E,i;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$.
   - *(d)* $\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$ *implies* $\Sigma;\Psi;E;\Gamma,J \xrightarrow{\nu} s \circ [u_1,u_2]$.
2. *(Contraction)* $\Sigma;\Psi;E;\Gamma,J,J \xrightarrow{\nu} s \circ [u_1,u_2]$ *implies* $\Sigma;\Psi;E;\Gamma,J \xrightarrow{\nu} s \circ [u_1,u_2]$.

Further the derivation in the consequent of each statement has a depth no more than that of the antecedent.

Proof. By separate induction on the given derivation for each property. □

Theorem 4.9 (Instantiation)**.** $\Sigma,x{:}\sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$ *and* $\Sigma \vdash t:\sigma$ *imply* $\Sigma;\Psi[t/x];E[t/x];\Gamma[t/x] \xrightarrow{\nu[t/x]} s[t/x] \circ [u_1[t/x],u_2[t/x]]$

Proof. By induction on the derivation of $\Sigma,x{:}\sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$. □

Theorem 4.10 (View subsumption)**.** *Suppose the following hold:*

1. $\Sigma;\Psi;E;\Gamma \xrightarrow{\nu} s \circ [u_1,u_2]$
2. $\nu = k_0,u_b,u_e$

3. $\Sigma; \Psi \models k_0 \succeq k_0'$, $\Sigma; \Psi \models u_b \leq u_b'$, *and* $\Sigma; \Psi \models u_e' \leq u_e$.

4. $\nu' = k_0', u_b', u_e'$

Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu'} s \circ [u_1, u_2]$ *by a derivation of smaller or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. There is only one interesting case which is shown below.

Case. $$\dfrac{\nu = k_0, u_b, u_e \qquad \Sigma; \Psi \models u_1' \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2' \qquad \Sigma; \Psi \models k \succeq k_0}{\Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]}\text{claims}$$

To show: $\Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'] \xrightarrow{\nu'} s \circ [u_1, u_2]$

1. $\Sigma; \Psi \models u_b \leq u_b'$ (Assumption 3)
2. $\Sigma; \Psi \models u_1' \leq u_b$ (Premise)
3. $\Sigma; \Psi \models u_1' \leq u_b'$ ((C-trans-time) from §4.2.1 on 1,2)
4. $\Sigma; \Psi \models u_e' \leq u_e$ (Assumption 3)
5. $\Sigma; \Psi \models u_e \leq u_2'$ (Premise)
6. $\Sigma; \Psi \models u_e' \leq u_2'$ ((C-trans-time) from §4.2.1 on 4,5)
7. $\Sigma; \Psi \models k_0 \succeq k_0'$ (Assumption 3)
8. $\Sigma; \Psi \models k' \succeq k_0$ (Premise)
9. $\Sigma; \Psi \models k' \succeq k_0'$ ((C-trans-prin) from §4.2.1 on 7,8)
10. $\Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'], r \circ [u_1', u_2'] \xrightarrow{\nu'} s \circ [u_1, u_2]$ (i.h. on 1st premise)
11. $\Sigma; \Psi; E; \Gamma, k' \text{ claims } s \circ [u_1', u_2'] \xrightarrow{\nu'} s \circ [u_1, u_2]$ (Rule (claims) on 10,3,6,9)

□

Theorem 4.11 (Time subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$
3. $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_n, u_m]$.

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. The proof appeals to Theorem 4.10 and a lemma about substitution of constraints. See Theorem B.4 in Appendix B for details of the lemma as well as some of the interesting cases of the proof. □

Theorem 4.12 (Admissibility of cut)**.** *The following two properties hold:*

1. *Suppose that*

   (a) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *and*

   (b) $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u'_1, u'_2]$

   Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u'_1, u'_2]$.

2. *Suppose that*

   (a) $\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]$

   (b) $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u'_1, u'_2]$

   Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u'_1, u'_2]$.

Proof. By a simultaneous lexicographic induction, first on the size of the cut formula s, then on the order (2) > (1), and finally on the depths of the two given derivations, as in prior work [43, 54, 113]. See Theorem B.6 in Appendix B for some of the cases of the proof. □

Theorem 4.13 (Identity)**.** *If* $\Sigma; \Psi \models u_1 \leq u'_1$ *and* $\Sigma; \Psi \models u'_2 \leq u_2$*, then* $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} s \circ [u'_1, u'_2]$.

Proof. By induction on s. The base case where s is an uninterpreted atom follows immediately from rule (init) in Figure 4.4. The base case where s is an interpreted atom is shown below. The third base case where s is a constraint is similar to it. All the remaining cases of the proof follow prior work on η logic [54].

Case. $s = i$. To show: $\Sigma; \Psi; E; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} i \circ [u'_1, u'_2]$.

1. $\Sigma; E, i \models i$ ((S-hyp) from §4.2.1)
2. $\Sigma; \Psi; E, i; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} i \circ [u'_1, u'_2]$ (Rule (interL) on 1)
3. $\Sigma; \Psi; E; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} i \circ [u'_1, u'_2]$ (Rule (interR) on 2)

□

4.2.6 Equivalence of Proof Systems

Despite differences in inference rules, the natural deduction system and the sequent calculus for BL establish exactly the same hypothetical judgments. This is formalized in the following theorem, which we prove by simulating the rules of each inference system in the other. Showing that each rule of the sequent calculus can be simulated in natural deduction is relatively straightforward – the right rules correspond to introduction rules directly, and left rules are easily simulated using elimination rules, together with the substitution principle (Theorem 4.5) in some cases.[5] Conversely, for simulating elimination rules of natural deduction in the sequent calculus, we appeal to admissibility of cut (Theorem 4.12) and identity (Theorem 4.13).

Theorem 4.14 (Equivalence)**.** *The following are equivalent.*

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *in the sequent calculus.*
2. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *in natural deduction.*

Proof. See Appendix B, Theorem B.7. □

4.3 Use of BL in PCFS

As mentioned in §2, the logic BL is used to express authorization policies in the file system PCFS and to enforce them. This section discusses briefly how files, principals, permissions, time points, and policy rules are represented concretely in the logic, what judgments need to proved in order to obtain access, and how PCFS enforces dynamism in policies that are either time sensitive or rely on system state.

Representation of principals, time, files, and permissions. Although the theory of BL discussed in this chapter is agnostic to the concrete representation of terms, from the perspective of an implementation, making this choice is important. In PCFS, principals are represented in one of two ways: either as symbolic constants, which may be added to a special declarations file that is protected by the back end (see §7 for details), or by their Linux user ids. In practice, the former representation is used for principals that do not correspond to any real users (e.g., organizational roles), while the latter is used for principals that do (e.g., users that run programs and access files). Access permissions are given on a per-file or per-directory basis to real users.

In PCFS, the clock of the reference monitor (file system back end) is the authority on time; all time points in logical formulas and procaps refer to this clock. Ground time points are represented in absolute form as YYYY:MM:DD:hh:mm:ss that can be abbreviated to YYYY:MM:DD when hh:mm:ss are 00:00:00. The parser for BL converts all time points to integers that measure seconds from a fixed reference point. The exact reference point is irrelevant for all practical purposes, but it should be noted that the precision of time used

[5] It is possible to simulate some rules of the sequent calculus rules in natural deduction by using the rules ($\supset$I) and ($\supset$E) in place of substitution. See, for example, Proposition 2.4 in [16].

in PCFS is one second. Fractions of time beyond this precision are rounded down whenever the system clock is read.

In addition to principal and time, the implementation of BL in PCFS supports two additional sorts: file (for file and directory names) and perm (for permissions). Ground files and directories are represented by their full path names relative to the path where PCFS is mounted. Thus, if PCFS is mounted at `/path/to/mountpoint`, then the file `/foo/bar` in any BL formula refers to the file `/path/to/mountpoint/foo/bar` in the file system. Making the path names relative to the mount point has the advantage that the mount point can be changed without having to change existing policy rules, proofs, and procaps. It has the disadvantage that without some explicit naming convention, there may be confusion among the policy rules of different PCFS file systems on the same server.

The sort of permissions, perm, contains five ground elements, which correspond to the five permissions that PCFS uses: read, write, execute, identity, and govern. Permissions read and write are needed to read and change a file or directory respectively, whereas execute is needed to read meta-data. The remaining two permissions, identity and govern, are described in §7.

Interpreted Predicates. PCFS natively supports two interpreted predicates in BL: **owner** f k, which means that file f has owner k, and **has_xattr** f a v, which means that file f has value v for the extended attribute `user.#pcfs.`a.[6] Both file ownership as well as extended attributes beginning with the prefix `user.#pcfs.` are specially protected by PCFS; the permission govern is needed to change them (details are in §7). It is expected that only trusted users will be given this permission. As a result, file ownership and extended attributes starting with the prefix `user.#pcfs.` can be used to classify files in a secure manner, and the interpreted predicates **owner** and **has_xattr** can be used to reference them in policy rules, as illustrated in the example in §4.3.3.

Support for other interpreted predicates can be added to BL's implementation in PCFS through a programming API provided for this purpose. As a convention, we write interpreted predicates in **boldface**.

Arithmetic on Time and Constraints. The implementation of BL in PCFS supports simple arithmetic over time points through a new sort exp whose elements are denoted e. exp includes all elements of time via a coercing function symbol and in addition includes terms of the forms $e_1 + e_2$, $e_1 - e_2$, `max`(e_1, e_2), and `min`(e_1, e_2). The terms in exp are *interpreted* via the new constraint form `is` u e, which means that the simplification of e in the usual arithmetic sense equals u. This idea of making simplification of arithmetic expressions explicit via the constraint form `is` u e is borrowed from logic programming in languages like Prolog. It should be noted that `is` u e is different from term equality in many other logics, because BL does not allow implicit substitution of e for u in judgments even if `is` u e holds. Nonetheless `is` u e is very useful for representing many policies of interest in BL, including those in §8.

[6] An extended attribute is a meta-data field on a file or directory whose value can be set by users. Many file systems including XFS, JFS, ext3, and PCFS support extended attributes.

4.3.1 Policies and Authorizations

Following past work on proof-carrying authorization, we assume in PCFS that formulas are established *a priori* by digitally signed certificates. In PCFS, it is assumed that every digitally signed certificate is valid during an interval of time, which is written inside the certificate before the latter is signed. This is consistent with what common certificate schemes like X.509 and PGP allow. In general, if principal k creates a certificate that is valid during the interval $[u_1, u_2]$ and contains the formula s, then this is represented in BL as the hypotheses k claims $s \circ [u_1, u_2]$. (Alternatively, we could have chosen one of the equivalent representations (k says s) $\circ$ $[u_1, u_2]$ and (k says s) @ $[u_1, u_2]$ $\circ$ $[-\infty, +\infty]$, but the representation we use is most convenient, since it is in some sense the simplest.)

A judgment k claims $s \circ [u_1, u_2]$ established by a signed certificate is called a *policy rule*, and a collection of such judgments is called a *policy*. The top level hypotheses Γ in all proof search and verification problems in BL are always a policy. PCFS also requires that a unique name be provided for each policy rule in the certificate that establishes the rule; this name is used to refer to the rule in proofs (§5).

What should be proved? PCFS assumes the existence of one distinguished principal, symbolically denoted admin, who has the ultimate authority on access. The actual identity of admin is provided to PCFS through a configuration file, which we discuss in §7. In order to get permission η on file f at time u, user k must prove that the policy in effect entails the defined basic judgment $\mathtt{auth}(k, f, \eta, u)$, where:

$$\mathtt{auth}(k, f, \eta, u) \triangleq \mathsf{admin}\ \mathsf{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [u, u]$$

$\mathtt{may}$ is a fixed uninterpreted predicate taking three arguments, and u is the time of access. (In PCFS, "time of access" refers to the time at which access checks for a file system call are initiated.) $[u, u]$ is a singleton set containing exactly the time point u. More precisely, it must be established that $\Sigma; \cdot; E; \Gamma \vdash^{\nu} \mathtt{auth}(k, f, \eta, u)$, where:

- Σ is the sorting in effect. It is specified through a signature file that is protected by PCFS (§7).
- E is the state of the file system at the time u.
- Γ is the policy, evidenced by digitally signed certificates as described earlier.
- ν is a view made of three fresh constants.

It can easily be seen that establishing $\Sigma; \cdot; E; \Gamma \vdash^{\nu} \mathtt{auth}(k, f, \eta, u)$ is equivalent to establishing $\Sigma; \cdot; E; \Gamma \vdash^{\nu_0} \mathtt{may}\ k\ f\ \eta \circ [u, u]$, with $\nu_0 = \mathsf{admin}, u, u$ whenever Γ is a policy.

4.3.2 Policy Enforcement

Although the use of the time interval $[u, u]$ in the judgment to be proved for authorization takes into account dependence of the policy on explicit time, it results in a practical problem:

how is the time of access u to be determined to the precision of a second at the time that a proof is constructed or verified, both of which happen prior to access in PCFS (§2)? Indeed, determining u to such high precision in advance of access is impossible in most settings. Fortunately, the time subsumption principle (Theorems 4.4 and 4.11) can be used to alleviate the problem in a reasonable way. Instead of constructing a proof of admin says (may k f η) $\circ$ $[u, u]$, a principal desirous of access may construct a proof of admin says (may k f η) $\circ$ $[u_1, u_2]$ where $[u_1, u_2]$ is a time interval that contains u. If this succeeds then the time subsumption principle guarantees that admin says (may k f η) $\circ$ $[u, u]$ is also provable, hence the former proof also witnesses the provability of the latter. In fact, in §5 we extend Theorem 4.4 to show that the *same proof term* which proves admin says (may k f η) $\circ$ $[u_1, u_2]$ also proves admin says (may k f η) $\circ$ $[u, u]$. Consequently, the exact time of access u need not be known at the time of proof construction; only a rough estimate of its range suffices. The proof verifier extracts from the proof the interval of time over which it is valid and writes the interval into the procap it generates. The back end of PCFS ensures that the time of access u is in this interval. The process of extraction of the time interval from a proof is explained in §5 and it is also shown formally that the process is sound.

A related problem arises for the state of the system E – how can the state at the time of access be estimated during proof construction or verification? To address this problem, the proof search tool in PCFS requires the user to provide selective input about expected state (§6), and the verification tool simply writes every interpreted atom it encounters in a proof to the procap it outputs. The back end then checks all such interpreted atoms at the time of access in the prevailing state (§5).

4.3.3 Example: Course Administration

We illustrate the use of BL through a simple example that expresses in the logic a policy for access to class homework directories in a hypothetical university UV. Assume that UV provides all class instructors storage space on a central server that can be used to collect homeworks from students. A principal called registrar decides the instructor of each class as well its teaching assistants (TAs) and students, expressed formally by the predicates (is-instructor k class), (is-ta k class), and (is-student k class) respectively. A separate principal diradmin assigns directories on the central server to classes; the predicate (is-dir dir class) means that directory dir has been assigned to class. Each directory is assumed to have an extended attribute user.#pcfs.state which determines who is allowed to read and write the directory. Possible values of this extended attribute are:

A. prep, meaning that the contents of the directory are being prepared. In this state only the TAs have read and write access to the directory.

B. submission, meaning that the homework in the directory is active. In this state all registered students can read and write the directory.[7]

[7] We side-step the issue of having a separate homework submission directory for each student, so strictly speaking, in this example, students will be able to read and write each others' homeworks. This can be easily avoided by modifying the formalization.

C. done, meaning that homework submission to the directory has been closed. In this state only the TAs have read access to the directory.

The value of the attribute user.#pcfs.state can only be changed by the instructor of the class to which the directory is assigned. This instructor always has both read and write permissions to the directory. Assuming that the principal admin has final control over determining access to directories on the central server, and that (may k d η) means that principal k has permission η on directory d, the policy rules for access are shown in rules (1)–(8) in Figure 4.6.

Rule (1) states that the instructor k of a class l may read any directory d associated with the class. The annotation $\circ\ [-\infty, +\infty]$ on this rule as well as on others means that the rule applies at all points of time. Rule (2) is similar, except that it authorizes write access. Rule (3) allows a TA k of class l to read a directory d associated with the class if the extended attribute user.#pcfs.state on the directory has been set to prep. Rule (4) is similar; it allows write access to TAs. Rules (5) and (6) allow students to read and write directories in the state submission. Rule (7) allows TAs to read directories in state done. The salient point to observe in rules (3)–(7) is the use of the interpreted predicate **has_xattr**, which guarantees the properties mentioned in (A)–(C) above.

Rule (8) allows the instructor of a class the authority to change the extended attributes of any directory associated with the class (and hence influence which of the rules (3)–(7) will apply to the directory), by giving her the govern permission. An important observation is that this authorization policy does not restrict the instructor's use of the govern permission to setting the attribute user.#pcfs.state in the specific order prep $\rightarrow$ submission $\rightarrow$ done. Instead we trust the instructor to follow this protocol correctly.

As a specific instance of the use of this policy, let us assume that Alice is instructor for class cs101 from August 20, 2009 to December 20, 2009. This would be established by a certificate from registrar, who would constrain its validity to exactly this interval of time. In BL, this certificate would be reflected as the judgment (9) in Figure 4.6. Further suppose that Terence is appointed TA for cs101 for the period September 01, 2009 to September 30, 2009. This may be represented by judgment (10). Finally assume that a directory cs101dir has been assigned to the class for the latter's duration (judgment (11)), and that on September 15, 2009, the attribute user.#pcfs.state on the directory cs101dir has value prep.

Then, it is quite easy to prove using the rules of the sequent calculus that the policy rules (4), (10), and (11) from Figure 4.6 entail the following judgment for any time point u in the interval September 01, 2009 – September 30, 2009, provided that the attribute user.#pcfs.state on cs101dir is set to prep, thus allowing Terence to write cs101dir at any such time u.

$$(\text{admin claims } (\texttt{may } \text{Terence cs101dir write})) \circ [u, u]$$

What is more interesting here is that the judgment cannot be proved if either u is not in the interval September 01, 2009 – September 30, 2009, or the attribute user.#pcfs.state on cs101 does not have the value prep. This illustrates how, through its combination of interpreted predicates, constraints, and explicit time, BL is able to express dynamic policies that rely on both system state as well as time.

General rules:

admin claims $\forall k, d, l.$
(1) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-instructor}\ k\ l))) \supset \texttt{may}\ k\ d\ \text{read}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(2) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-instructor}\ k\ l))) \supset \texttt{may}\ k\ d\ \text{write}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(3) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-ta}\ k\ l)) \wedge (\textbf{has\_xattr}\ d\ \text{state prep})) \supset \texttt{may}\ k\ d\ \text{read}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(4) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-ta}\ k\ l)) \wedge (\textbf{has\_xattr}\ d\ \text{state prep})) \supset \texttt{may}\ k\ d\ \text{write}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(5) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-student}\ k\ l)) \wedge (\textbf{has\_xattr}\ d\ \text{state submission})) \supset \texttt{may}\ k\ d\ \text{read}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(6) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-student}\ k\ l)) \wedge (\textbf{has\_xattr}\ d\ \text{state submission})) \supset \texttt{may}\ k\ d\ \text{write}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(7) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-ta}\ k\ l)) \wedge (\textbf{has\_xattr}\ d\ \text{state done})) \supset \texttt{may}\ k\ d\ \text{read}) \circ [-\infty, +\infty]$

admin claims $\forall k, d, l.$
(8) $(((\text{diradmin says } (\texttt{is-dir}\ d\ l)) \wedge (\text{registrar says } (\texttt{is-instructor}\ k\ l))) \supset \texttt{may}\ k\ d\ \text{govern}) \circ [-\infty, +\infty]$

Rules specific to an instance:

(9) registrar claims (`is-instructor` Alice cs101) $\circ$ [2009:08:20, 2009:12:20]

(10) registrar claims (`is-ta` Terence cs101) $\circ$ [2009:09:01, 2009:09:30]

(11) diradmin claims (`is-dir` cs101dir cs101) $\circ$ [2009:08:20, 2009:12:20]

Figure 4.6: Policy rules for access to class directories

A large case study on the use of BL for controlling access to classified information that is based on ideas similar to those in this example is presented in §8. Other examples of the use of explicit time, but not interpreted predicates, in the context of authorization may be

found in prior joint work of the author [54].

4.4 Justification for the Use of Time Points in BL Views

In generalizing the logic from BL_S to BL we have also generalized views from being principals k_0 to triples k_0, u_b, u_e. The question is whether this generalization is necessary. More precisely, can we systematically erase the time points u_b, u_e from views in all rules of the sequent calculus (Figures 4.4 and 4.5), and work with the resulting logic?

From the perspective of proof theory there is no problem with this new logic. Its proof theory is simpler than that of BL and analogues of theorems of §4.2.5 are admissible in it. The problem with the logic lies in its expressiveness – it admits the following sequent (which BL does not in general).

$$\Sigma; \cdot; \cdot; k \text{ claims } (s \mathbin{@} [u_1, u_2]) \circ [u_1', u_2'] \xrightarrow{k_0} k \text{ says } s \circ [u_1, u_2]$$

Put more succinctly, the formulas $(k \text{ says } (s \mathbin{@} [u_1, u_2])) \mathbin{@} [u_1', u_2']$ and $(k \text{ says } s) \mathbin{@} [u_1, u_2]$ are equivalent in the logic. As an example of the consequences of this expressiveness, suppose that principal admin signs the formula (may Alice foo.txt read) @ [2009:01:01, 2009:12:31], i.e. a formula that allows Alice access throughout 2009, and puts it in a certificate that is valid during the interval [2009:01:01, 2009:06:30]. Note that the certificate itself expires on June 30, 2009. According to the description in §4.3, this certificate would be reflected in the logic as the hypothesis:

$$\text{admin claims } ((\texttt{may Alice foo.txt read}) \mathbin{@} [2009{:}01{:}01, 2009{:}12{:}31]) \circ [2009{:}01{:}01, 2009{:}06{:}30]$$

Now we ask the question: Given this and only this hypothesis, should Alice be allowed to read foo.txt on September 01, 2009 at 00:00:00 hours? Or equivalently, should this hypothesis entail the judgment

$$\text{admin claims } (\texttt{may Alice foo.txt read}) \circ [2009{:}09{:}01, 2009{:}09{:}01]$$

If we use BL, the answer would be no, as can easily be proved using the sequent calculus. However, if we were to use the modified logic, the answer would be yes – the required entailment is an instance of the sequent shown earlier, followed by one use of time subsumption.

The question then is which of the two answers is correct – should Alice be allowed the access or not? In this case it seems that BL's answer is the correct one. Since admin's certificate expires on June 30, 2009 and nothing in the policy explicitly allows the use of an expired certificate, a proof constructed from the admin's certificate should not be acceptable on September 01, 2009. It is for this reason – to prevent implicit use of expired certificates and to take into account the validity of a certificate even if the content of the certificate mentions another time interval – that we choose to keep the time points u_b, u_e in the views in BL. In settings where this is not desirable, the other logic obtained by dropping time points from views may be more appropriate.

4.5 Proof Normalization

The subject of this section is orthogonal to the rest of the thesis and the disinclined reader may skip it without a break in continuity.

We show that every natural deduction proof can be reduced to a proof in canonical form. The latter are a subclass of proofs. Reduction of a proof to a canonical form is called proof normalization. Although not directly useful in PCFS, it is an important proof-theoretic result. Canonical proofs also lead to bidirectional proof verification, which is used in PCFS (§5). An interesting aspect of our proof of the existence of canonical proofs is the use of the sequent calculus and its equivalence to natural deduction (Theorem 4.14) instead of the usual approach of defining a proof rewrite system such as β-reduction, and showing that it always terminates. The technical material in this section is a generalization of similar work for other logics [114, 119, 145].

What is a canonical proof? By a canonical proof we mean a natural deduction proof that has no β-redexes and to which commuting conversions have been applied to the maximum possible extent. Both β-redexes and commuting conversions are informally explained below, and canonical proofs are formally defined later.

A β-redex is a locus in a proof where the principal formula of an elimination rule is established using an introduction rule. For example, consider a proof which ends as shown below. This proof has a β-redex since the formula k says s is introduced using the rule (saysI) and immediately eliminated using the rule (saysE). Due to the presence of this β-redex the proof is not canonical.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Sigma; \Psi; E; \Gamma| \vdash^{k,u_1,u_2} s \circ [u_1, u_2]\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } s \circ [u_1, u_2]}\,\mathsf{saysI} \qquad \begin{array}{c}\mathcal{E}\\ \Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2']\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} r \circ [u_1', u_2']}\,\mathsf{saysE}$$

This β-redex can be eliminated by applying Theorem 4.6 to the derivations $\mathcal{D}$ and $\mathcal{E}$.

A commuting conversion is a proof transformation that allows let-like elimination rules, i.e. all elimination rules except ($\wedge$E$_1$), ($\wedge$E$_2$), ($\supset$E), and ($\forall$E), to be pushed outside of other elimination rules. For instance consider the following proof in which the rule (saysE) is used to prove a judgment that is principal in the rule ($\wedge$E$_1$).

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } r \circ [u_1', u_2']\end{array} \qquad \begin{array}{c}\mathcal{E}\\ \Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'] \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]}\,\wedge\mathrm{E}_1}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2]}\,\mathsf{saysE}$$

A commuting conversion may be applied to rotate the two rules, resulting in the following proof.

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } r \circ [u_1', u_2']\end{array} \qquad \dfrac{\begin{array}{c}\mathcal{E}\\ \Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'] \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]\end{array}}{\Sigma; \Psi; E; \Gamma, k \text{ claims } r \circ [u_1', u_2'] \vdash^{\nu} s_1 \circ [u_1, u_2]}\,\wedge\mathrm{E}_1}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2]}\,\mathsf{saysE}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow \qquad \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1', u_2'] \Uparrow} \Downarrow\Uparrow$$

$$\frac{}{\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow} \text{hyp}$$

$$\frac{\nu = k, u_b, u_e \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \qquad \Sigma; \Psi \models k' \succeq k}{\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow} \text{claims}$$

$$\frac{\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2] \Uparrow} \mathsf{saysI}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2] \Downarrow \qquad \Sigma; \Psi; E; \Gamma, k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow} \mathsf{saysE}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} (s \ @ \ [u_1, u_2]) \circ [u_1', u_2'] \Uparrow} @\text{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \ @ \ [u_1, u_2] \circ [u_1', u_2'] \Downarrow \qquad \Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1'', u_2''] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1'', u_2''] \Uparrow} @\text{E}$$

$$\frac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} c \circ [u_1, u_2] \Uparrow} \text{consI}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} c \circ [u_1, u_2] \Downarrow \qquad \Sigma; \Psi, c; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow} \text{consE}$$

$$\frac{\Sigma; E \models i}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} i \circ [u_1, u_2] \Uparrow} \text{interI}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} i \circ [u_1, u_2] \Downarrow \qquad \Sigma; \Psi; E, i; \Gamma \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2'] \Uparrow} \text{interE}$$

Figure 4.7: BL: Canonical and atomic proofs, part 1

Formal definition. Formally, we characterize canonical proofs using two judgments $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ and $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ that are defined by the mutually inductive rules in Figures 4.7 and 4.8. The judgment $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ means that $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ has a *canonical proof* whereas $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ means that $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ has what we call an *atomic proof*. Atomic proofs are an auxiliary class of proofs that we need to define canonical proofs.

The rules defining these judgments are similar to those of natural deduction (Figures 4.2 and 4.3), and are also named similarly. The obvious differences are: (a) One of the symbols $\Uparrow$ and $\Downarrow$ is placed at the end of each hypothetical judgment, (b) There is a new rule ($\Downarrow\Uparrow$)

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \Uparrow \qquad \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2] \Uparrow} \wedge\mathrm{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2] \Downarrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \Downarrow} \wedge\mathrm{E}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2] \Downarrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2] \Downarrow} \wedge\mathrm{E}_2$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2] \Uparrow} \vee\mathrm{I}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2] \Uparrow} \vee\mathrm{I}_2$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \vee s_2 \circ [u_1, u_2] \Downarrow \\ \Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2] \vdash^{\nu} s' \circ [u'_1, u'_2] \Uparrow \qquad \Sigma; \Psi; E; \Gamma, s_2 \circ [u_1, u_2] \vdash^{\nu} s' \circ [u'_1, u'_2] \Uparrow\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u'_1, u'_2] \Uparrow} \vee\mathrm{E}$$

$$\frac{}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \top \circ [u_1, u_2] \Uparrow} \top\mathrm{I} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \bot \circ [u_1, u_2] \Downarrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u'_1, u'_2] \Uparrow} \bot\mathrm{E}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \Uparrow} \supset\mathrm{I}$$

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \Downarrow \\ \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u'_1, u'_2] \Uparrow \qquad \Sigma; \Psi \models u_1 \leq u'_1 \qquad \Sigma; \Psi \models u'_2 \leq u_2\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u'_1, u'_2] \Downarrow} \supset\mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \forall x{:}\sigma.s \circ [u_1, u_2] \Uparrow} \forall\mathrm{I} \qquad \frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \forall x{:}\sigma.s \circ [u_1, u_2] \Downarrow \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s[t/x] \circ [u_1, u_2] \Downarrow} \forall\mathrm{E}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s[t/x] \circ [u_1, u_2] \Uparrow \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \exists x{:}\sigma.s \circ [u_1, u_2] \Uparrow} \exists\mathrm{I}$$

$$\frac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \exists x{:}\sigma.s \circ [u_1, u_2] \Downarrow \qquad \Sigma, x{:}\sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u'_1, u'_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u'_1, u'_2] \Uparrow} \exists\mathrm{E}$$

Figure 4.8: BL: Canonical and atomic proofs, part 2

which coerces from atomic proofs to canonical proofs, and (c) In the rules (hyp) and (claims) the principal hypothesis and the conclusion are true on the same time interval. Change (c) is motivated by a desire to be able to bidirectionally check canonical proofs. This should become clear in §5. Observe the following:

1. Every introduction rule results in a canonical proof, and its principal premise requires a canonical proof.
2. Every let-like elimination rule results in a canonical proof.
3. The rules (hyp), (claims), ($\wedge\mathrm{E}_1$), ($\wedge\mathrm{E}_2$), ($\supset\mathrm{E}$), and ($\forall\mathrm{E}$) result in atomic proofs.

4. The principal premise of every elimination rule must be atomic.

5. The rule ($\Downarrow\Uparrow$) allows atomic proofs to be treated as canonical, but there is no rule to coerce canonical proofs to atomic proofs.

(1), (4), and (5) imply that a canonical proof has no β-redexes. Further, (2) and (4) imply that in every canonical proof commuting conversions have been fully applied.

Properties of canonical and atomic proofs. The most obvious property of canonical and atomic proofs is that each hypothetical judgment that has an atomic or canonical proof also has a natural deduction proof. This is fairly easy to prove by induction on atomic and canonical proofs.

Theorem 4.15 (Injection)**.** *The following hold.*

1. *If* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ *then* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$.

2. *If* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ *then* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$.

Proof. By simultaneous induction on given derivations, and case analysis of the last rules in them. The cases of (hyp) and (claims) rely on property (C-refl-time) from §4.2.1, whereas the case of ($\Downarrow\Uparrow$) uses Theorem 4.4. □

The following analogue of the view subsumption (Theorem 4.3) is straightforward.

Theorem 4.16 (View subsumption)**.** *Suppose the following hold:*

1. $\nu = k_0, u_b, u_e$ *and* $\nu' = k_0', u_b', u_e'$

2. $\Sigma; \Psi \models k_0 \succeq k_0'$, $\Sigma; \Psi \models u_b \leq u_b'$, *and* $\Sigma; \Psi \models u_e' \leq u_e$

Then,

A. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu'} s \circ [u_1, u_2] \Downarrow$ *by a derivation of shorter or equal depth.*

B. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu'} s \circ [u_1, u_2] \Uparrow$ *by a derivation of shorter or equal depth.*

Proof. By simultaneous induction on given derivations in A and B, and case analysis of the last rule. The case where the derivation in A ends in (claims) uses the assumptions (C-trans-time) and (C-trans-prin) from §4.2.1, as in the proof of Theorem 4.3. □

Next we consider the analogue of time subsumption (Theorem 4.4) for canonical proofs. Since in the rules (hyp) and (claims) of Figure 4.7 we require that the time interval in the hypothesis and that in the conclusion match, time subsumption does not hold for atomic proofs. However, the rule ($\Downarrow\Uparrow$) allows subsumption with respect to time intervals, as a result of which canonical proofs admit time subsumption, as formalized by the following theorem.

Theorem 4.17 (Time subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$
2. $\Sigma; \Psi \models u_1 \leq u_n$ *and* $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m] \Uparrow$

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ and case analysis of its last rule. The case where the derivation ends in rule (saysI) uses view subsumption (Theorem 4.16). In the case where the derivation ends in rule ($\supset$I), a lemma about substitution of constraints is needed. See Appendix B, Theorem B.9 for details. □

Both atomic and canonical proofs are closed under substitution by atomic proofs (next theorem). However, substitution of a canonical proof for a hypothesis may result in creation of a β-redex or a new commuting conversion, and hence atomic and canonical proofs are not closed under substitution by canonical proofs.

Theorem 4.18 (Substitution)**.** *Suppose* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$. *Then the following hold.*

1. $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} r \circ [u'_1, u'_2] \Downarrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} r \circ [u'_1, u'_2] \Downarrow$.
2. $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} r \circ [u'_1, u'_2] \Uparrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} r \circ [u'_1, u'_2] \Uparrow$.

Proof. By simultaneous induction on derivations given in (1) and (2), and case analysis of their last rules. □

Proof Normalization. Finally, we show that if a hypothetical judgment has a natural deduction proof, then it also has a canonical proof. By Theorem 4.14 we know that every natural deduction proof can be simulated in the sequent calculus. Given this fact, it suffices to show that every provable sequent has a canonical proof.

Theorem 4.19 (Normalization)**.** *Suppose* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *in natural deduction. Then* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$.

Proof. By Theorem 4.14 it suffices to show that $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ implies $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$. This can be proved by induction on the depth of the given sequent calculus proof and a case analysis of its last rule, making considerable use of Theorem 4.18. See Appendix B, Theorem B.10 for details of some representative cases. □

Since both the simulation of natural deduction in the sequent calculus (Theorem 4.14) and the simulation of the sequent calculus by canonical proofs (Theorem 4.19) are established *constructively*, there is an algorithm that converts a natural deduction proof to a canonical proof. This algorithm may be obtained by converting the inductive cases in the proofs of the two theorems to clauses of declarative programs. This algorithm is easy but tedious to describe, and since it does not provide any new insights, we omit its details.

4.6 Relation between BL_S and BL

So far, we have not described how the two logics BL_S (§3) and BL are related. Using the sequent calculus for BL it is easy to show that axiom (C) of BL_S (§3.1.1) is not admissible in BL, so BL is not a conservative extension of BL_S. What, then, is the relation between the two logics? In this section we present two results in this regard. First we show that the translation from BL to BL_S that erases time intervals, constraints, and interpreted atoms maps provable hypothetical judgments to provable ones. In this sense BL is a generalization of BL_S. Second we show that there is a simple embedding of BL_S in BL that preserves provability and unprovability of sequents.

Translation from BL to BL_S. Figure 4.9 defines a translation $|\cdot|$ from BL to BL_S that maps constraints and interpreted atoms to $\top$ and erases @ connectives as well as suffixes $\circ\,[u_1, u_2]$. (To avoid having to translate sorts, we assume that time is a sort in BL_S but do not assume any specific properties of it.) As the following theorem states, the image of a sequent provable in BL is also provable in BL_S, if we assume that constraints of the form $k \succeq k'$ do not appear in Ψ, Γ, and the conclusion. This restriction is needed to incorporate the fact that the order $\succeq$ among principals in BL_S is statically fixed, which may not be the case in BL in general. For instance if principals k and k' are unrelated in BL_S, then even though $(k \succeq k') \supset ((k \;\mathsf{says}\; s) \supset (k' \;\mathsf{says}\; s))$ is provable in BL, its translation $\top \supset ((k \;\mathsf{says}\; \ulcorner s \urcorner) \supset (k' \;\mathsf{says}\; \ulcorner s \urcorner))$ may not be provable in BL_S. Along the same lines, the theorem assumes that $\Sigma'; \Psi' \models k \succeq k'$ in BL implies $\Sigma' \vdash k \succeq k'$ in BL_S, whenever Ψ' does not have $\succeq$ in it.

Theorem 4.20 (Soundness of translation)**.** *Suppose that the constraint constructor $\succeq$ does not appear in either Ψ, Γ, or s and further suppose that for every k, k', Σ' and Ψ' not containing $\succeq$, $\Sigma'; \Psi' \models k \succeq k'$ in BL implies $\Sigma' \vdash k \succeq k'$ in BL_S. Then provability of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ in BL implies provability of $|\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]|$ in BL_S.*

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. □

Embedding BL_S in BL. Figure 4.10 shows a translation $\ulcorner \cdot \urcorner$ from BL_S to BL that we claim is an embedding. The translation puts the suffix @ $[-\infty, +\infty]$ inside the condition of every implication and adds the suffix $\circ\,[-\infty, +\infty]$ to every basic judgment of BL_S. This remarkably simple translation preserves provability, as the following theorem states. The key observation in the proof of the theorem is that because of the suffix @ $[-\infty, +\infty]$ inside every implication, it can be ensured that all intervals on basic judgments in the hypotheses in the proof of a translated sequent remain $[-\infty, +\infty]$.

Theorem 4.21 (Correctness of embedding)**.** *Suppose that for every k, k', Σ', and Ψ' not containing $\succeq$, $\Sigma'; \Psi' \models k \succeq k'$ in BL if and only if $\Sigma' \vdash k \succeq k'$ in BL_S. Then, $\Sigma; \Gamma \xrightarrow{k_0} s$ is provable in BL_S if and only if $\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner$ is provable in BL.*

Formulas s

$$
\begin{array}{rcl}
|p| & = & p \\
|i| & = & \top \\
|c| & = & \top \\
|s_1 \wedge s_2| & = & |s_1| \wedge |s_2| \\
|s_1 \vee s_2| & = & |s_1| \vee |s_2| \\
|s_1 \supset s_2| & = & |s_1| \supset |s_2| \\
|\top| & = & \top \\
|\bot| & = & \bot \\
|\forall x{:}\sigma.s| & = & \forall x{:}\sigma.|s| \\
|\exists x{:}\sigma.s| & = & \exists x{:}\sigma.|s| \\
|k \text{ says } s| & = & k \text{ says } |s| \\
|s \mathbin{@} [u_1, u_2]| & = & |s|
\end{array}
$$

Basic Judgments J

$$
\begin{array}{rcl}
|s \circ [u_1, u_2]| & = & |s| \text{ true} \\
|k \text{ claims } s \circ [u_1, u_2]| & = & k \text{ claims } |s|
\end{array}
$$

Hypotheses Γ

$$
|J_1, \ldots, J_n| \quad = \quad |J_1|, \ldots, |J_n|
$$

Views ν

$$
|k_0, u_b, u_e| \quad = \quad k_0
$$

Sequents

$$
|\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]| \quad = \quad \Sigma; |\Gamma| \xrightarrow{|\nu|} |s|
$$

Figure 4.9: Translation $|\cdot|$ from BL to BL_S

Proof. The "if" direction follows from the observation that $|\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner| = \Sigma; \Gamma \xrightarrow{k_0} s$. So if $\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner$ is provable in BL, then by Theorem 4.20, $|\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner|$ is provable in BL_S, or equivalently, $\Sigma; \Gamma \xrightarrow{k_0} s$ is provable in BL_S.

Proof of the "only if" direction follows by an induction on the depth of the given BL_S derivation of $\Sigma; \Gamma \xrightarrow{k_0} s$ and a case analysis of its last rule. See Appendix B, Theorem B.11 for some of the interesting cases. □

Formulas s

$$
\begin{array}{rcl}
\ulcorner p \urcorner & = & p \\
\ulcorner s_1 \wedge s_2 \urcorner & = & \ulcorner s_1 \urcorner \wedge \ulcorner s_2 \urcorner \\
\ulcorner s_1 \vee s_2 \urcorner & = & \ulcorner s_1 \urcorner \vee \ulcorner s_2 \urcorner \\
\ulcorner s_1 \supset s_2 \urcorner & = & (\ulcorner s_1 \urcorner \mathbin{@} [-\infty, +\infty]) \supset \ulcorner s_2 \urcorner \\
\ulcorner \top \urcorner & = & \top \\
\ulcorner \bot \urcorner & = & \bot \\
\ulcorner \forall x{:}\sigma.s \urcorner & = & \forall x{:}\sigma.\ulcorner s \urcorner \\
\ulcorner \exists x{:}\sigma.s \urcorner & = & \exists x{:}\sigma.\ulcorner s \urcorner \\
\ulcorner k \ \mathsf{says}\ s \urcorner & = & k \ \mathsf{says}\ \ulcorner s \urcorner
\end{array}
$$

Basic Judgments J

$$
\begin{array}{rcl}
\ulcorner s \ \mathsf{true} \urcorner & = & \ulcorner s \urcorner \circ [-\infty, +\infty] \\
\ulcorner k \ \mathsf{claims}\ s \urcorner & = & k \ \mathsf{claims}\ \ulcorner s \urcorner \circ [-\infty, +\infty]
\end{array}
$$

Hypotheses Γ

$$
\ulcorner J_1, \ldots, J_n \urcorner = \ulcorner J_1 \urcorner, \ldots, \ulcorner J_n \urcorner
$$

Views k_0

$$
\ulcorner k_0 \urcorner = k_0, -\infty, +\infty
$$

Sequents

$$
\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner = \Sigma; \cdot; \cdot; \ulcorner \Gamma \urcorner \xrightarrow{\ulcorner k_0 \urcorner} \ulcorner s \urcorner \circ [-\infty, +\infty]
$$

Figure 4.10: Embedding $\ulcorner \cdot \urcorner$ from BL_S to BL

Example 4.22. We mentioned earlier that BL is not a conservative extension of BL_S because axiom (C) of BL_S – k says $((k \text{ says } s) \supset s)$ – is not admissible in BL. However, it is easy to prove using the sequent calculus of BL that the translation of (C), i.e. k says $(((k \text{ says } \ulcorner s \urcorner) \mathbin{@} [-\infty, \infty]) \supset \ulcorner s \urcorner)$ is provable in BL for every k and s.

4.7 Related Work

There is a significant amount of work related to BL, both in the area of proof theory, and in the area of declarative formalisms for representing authorization policies that depend on time and system state.

Differences from η logic. As mentioned in the introduction, the treatment of explicit time in BL is based on η logic, which was first published in joint work of DeYoung, the author, and Pfenning [54], and largely developed in DeYoung's undergraduate thesis [53]. Besides the fact that the treatment of says in η logic is derived from GP logic, whereas that in BL is based on BL_S, there are also several other minor differences between the logics. First, the interaction between says and time in BL is richer than it is in η due to the presence of views (see §4.4). Second, the treatment of constraints in BL is slightly more general than it is in η – the latter confines constraints to the constructs $c \wedge s$ and $c \supset s$. This is necessary because η logic also considers well-formedness of intervals embedded in formulas; for $[u_1, u_2]$ to be well-formed, $u_1 \leq u_2$ is a pre-requisite. Checking well-formedness necessitates collection of constraints statically from formulas, which is greatly eased if occurrences of constraints are restricted. In BL we elide this well-formedness check. BL's constraints can be recovered in η by defining the formula c as $c \wedge \top$, and conversely, well-formedness checks can be incorporated in BL without difficulty.

In terms of expressiveness, the embedding from GP logic to BL_S (§3.5.1) easily extends to an embedding from η logic to BL, so BL is at least as expressive as η logic. Further, for reasons mentioned in §3.1.2 exclusive delegation cannot be expressed easily in η logic, but can be expressed readily in BL.

Hybrid logics. A hybrid logic is a modal logic, the worlds of whose Kripke structures are made explicit in formulas. Hybrid logics include modal formulas of the form $s \mathbin{@} w$, which means that formula s is true at world w. $s \mathbin{@} [u_1, u_2]$ is a specific kind of hybrid connective, where the worlds have the structure of intervals on the integer line. Since the only properties of intervals used in BL and η logic are reflexivity and transitivity of interval containment, one may think of $s \mathbin{@} [u_1, u_2]$ as a hybrid modality over a Kripke structure whose worlds form a partial order. Such Kripke models have been used to interpret intuitionistic logic in the past (see for example, [30]). Indeed the somewhat unusual ($\supset$I) rule in Figure 4.3 corresponds to the definition of satisfaction of implication in Kripke models. In this sense, BL is related to a lot of existing work on hybrid logics, and in particular, intuitionistic hybrid logics [35, 122].

Constraints. As in η logic, integration of constraints and the proof system in BL is directly based on the work of Saranli and Pfenning [128] and that of Jia [84], both of which were in the context of linear logic. There has also been a significant amount of work on integrating constraint domains in logic programming languages. Since the latter line of work is not directly related to BL, we refer the reader to a survey for its details [82].

Within the context of authorization policies, a number of logic-based frameworks for expressing policies allow representation of constraints, e.g., [18, 23, 26, 94, 95]. The treatment of constraints in all these is similar to that in constrained logic programming. The author is unaware of any authorization logics with constraints prior to η logic.

Explicit Time. Frühwith's work on Temporal Annotated Constraint Logic Programming (TACLP) [62] is very closely related to BL's treatment of explicit time. Although not

intended to express authorization policies, TACLP contains a modality p `at` u, which means that the atomic formula p is true at time u. The notion corresponding to our connective s @ $[u_1, u_2]$ (called s *th* I in TACLP, with I being an interval) is then *defined* by first requiring that `at` commute with all other connectives including implication, and then defining s *th* I as $\forall u.\ ((u \in I) \supset (s$ `at` $u))$. The obvious differences between BL and TACLP are that the latter lacks `says` and is a logic programming language, not a full logic.

In place of the @ modality, declarative frameworks for expressing authorization policies often include an interpreted constant that represents the time at which an operation such as proof search or proof verification is performed. This approach has been adopted, among others, in the language SecPAL where the constant `currentTime`() reduces to the time of evaluation of the authorization query [23], and in work on proof-carrying authorization where the constant `localtime` evaluates to the time of access [18]. Irrespective of the name, policy expiration can be represented using this constant in a simple manner; in proof-carrying authorization, for instance, one may say $(u_1 < \texttt{localtime} < u_2) \supset s$ instead of s to force s to be usable only in the time interval (u_1, u_2). The limitation of this approach, as opposed to explicit time in BL, is that it does not work when formulas need to be proved on intervals of time other than [`localtime`, `localtime`]. For example, the policy rule in the paragraph "Anachronistic references" in §4.1.2 cannot be expressed with this constant alone, because the condition (`mayaccess` k f) in the implication needs to proved on the interval $[T, T]$, which is different from the time of access. More precisely, the approach works only if the policy rules do not have @ connectives nested inside other connectives.

Along similar lines, it is common in implementations of proof-carrying authorization to use extra-logical checks to enforce expiration of credentials since the logics used cannot represent time [18, 20]. Instead, it is checked in the reference monitor that the time of access lies in the intersection of the validities of all credentials used in the proof that authorizes access. In contrast, in logics like BL and η, the logic represents validities of certificates (§4.3), and proofs contain the same information, so these extra-logical checks are internalized into proof verification. PCFS goes a step further since information about time is extracted from proofs during proof verification and placed explicitly in procaps that are checked at the time of access (§5). Checking expiration of certificates directly is equivalent to the checks internalized in proof verification in BL if all @ connectives are at the top level. This was shown formally in prior work on η logic [54].

System state. Independent of the work in this thesis and concurrently with it, Schneider et al. have designed the Nexus Authorization Logic (NAL) [132] and implemented it in the Nexus operating system [142]. NAL includes support for interpreted predicates in a manner similar to that in BL – in the reference monitor certain predicates are verified using trusted decision procedures that may refer to the system state. The implementation of the Nexus operating system uses a mix of proof-carrying authorization and inference in reference monitors to enforce authorization policies written in NAL.

Several other logic-based frameworks for representing authorization policies [18, 23, 26, 94, 95] do not make a distinction between constraints and predicates interpreted on the state of the system, and consequently support system state implicitly as part of their support

for constraints. The distinction between constraints and predicates interpreted on system state is necessitated in BL because of the PCFS requirement to verify proofs in advance of access. This is not the case with any other logic-based authorization framework.

There has also been some work on declarative languages and logics in which authorization policies and *state transitions* can be represented simultaneously [22, 25, 55]. In such frameworks, the state is represented through uninterpreted predicates, whose transition rule(s) are also defined in the framework. This line of work is largely orthogonal to BL.

Chapter 5

BL Proof Terms, Their Verification, and Procaps

In §4 we explained BL's inference rules and the structure of BL's proofs. This chapter introduces proof terms, a compact representation of natural deduction proofs that is used in PCFS, and describes the procedure used for their verification. Whereas a description of proof terms and their verification for a logic may seem trivial in general, and perhaps not critical enough to merit an independent chapter in a thesis, there are at least three reasons why a thorough investigation of the same is justified here.

First, in BL we are interested in simultaneously minimizing annotations in proof terms to keep them small, and having a simple verification procedure whose implementation is efficient and easily trusted. It is well known that these two problems are at odds with each other. As extreme examples of the trade-off in intuitionistic logic, the Church-style of proof terms results in an easy linear-time verification procedure but requires that every bound variable be annotated with the judgment whose proof it represents; the Curry-style of proof terms, on the other hand, mandates no annotations but results in a difficult verification problem [44, 50]. In BL the trade-off is further complicated by a need to have the time subsumption principle, i.e. we want a proof term that witnesses $s \circ [u_1, u_2]$ to also witness $s \circ [u_1', u_2']$ whenever $[u_1', u_2'] \subseteq [u_1, u_2]$. (See §4.3.2 for an explanation of the importance of this principle.) Therefore, the Church-style of proof terms, which would allow a proof term to establish at most one judgment does not work for BL. Our final design of proof terms for BL is based on an adaptation of bidirectional proofs, where proof verification is combined with selective inference of judgments established by proof terms, and use of annotations is restricted to β-redexes [117]. Although not surprising, this choice breaks the one-to-one correspondence between natural deduction proofs and proof terms (e.g., the Curry-Howard isomorphism), so some effort is needed to show that proof terms cover all natural deduction proofs (§5.1.1). Further, we formally prove that the time subsumption principle as well as several other intuitive properties hold for proof terms (§5.1.2).

Second, the proof verification problem for BL is non-trivial because proofs depend on system state and time of use, both of which may change between the time of proof verification and the time of access in PCFS (§4.3.2). (Recall from §2 that proof verification

in PCFS happens in advance of file access to keep the latter efficient.) Consequently, for PCFS, we use a non-standard proof verification procedure that *does not check* parts of a proof that depend on either system state or time of use, but instead outputs them as conditions in the procap (capability) generated from the proof. The back end of PCFS then completes the proof verification by checking these conditions every time the procap is used for access. Explaining this proof verification procedure and *proving* that this two part checking results in accurate verification of proofs is one of the most important goals of this chapter (§5.2). In addition, the chapter also describes the abstract structure and checking of procaps (§5.2.3, §5.2.1), and discusses how policy revocation may be implemented in the PCFS architecture (§5.2.4).

Third, even though proof terms witness natural deduction proofs, the proof search tool included in PCFS constructs sequent calculus proofs. Consequently, we need to produce natural deduction proof terms from sequent calculus proofs, which although not difficult, is not entirely trivial (§5.3). We also explain proof terms for canonical proofs of BL and argue that all proof terms can be reduced to a canonical form (§5.4).

5.1 Bidirectional Proof Terms for BL

BL proof terms are divided into two mutually inductive syntactic categories – checkables and inferables. During proof verification it is *checked* that a checkable correctly establishes a given basic judgment from given hypotheses, whereas the basic judgment witnessed by an inferable is *inferred*. In this sense, the proof term system and verification are both bidirectional [117]. The syntax of proof terms is summarized below. There is one proof term constructor for every rule in natural deduction (the names of proof term constructors and natural deduction rules correspond to each other). In addition, each inferable is also a checkable, and there is a special constructor `check` that coerces checkables to inferables. `check` is the only constructor whose arguments mention BL formulas. The letters τ, π denote proof variables which are used to name hypotheses in a proof. The notation $\pi.\Xi$ means that the proof variable π is bound in the syntactic entity Ξ and may be subject to α-renaming; $x.\Xi$ is similar notation for binding the term variable x.

$$
\begin{array}{llll}
\text{Checkables} & V & ::= & R \mid \texttt{conjI}\ V_1\ V_2 \mid \texttt{disjI1}\ V \mid \texttt{disjI2}\ V \mid \texttt{disjE}\ R\ (\pi_1.V_1)\ (\pi_2.V_2) \\
& & & \mid \texttt{topI} \mid \texttt{botE}\ R \mid \texttt{impI}\ (x_1.x_2.\pi.V) \mid \texttt{forallI}\ (x.V) \mid \texttt{existsI}\ t\ V \\
& & & \mid \texttt{existsE}\ R\ (x.\pi.V) \mid \texttt{atI}\ V \mid \texttt{atE}\ R\ (\pi.V) \mid \texttt{saysI}\ V \\
& & & \mid \texttt{saysE}\ R\ (\pi.V) \mid \texttt{consI} \mid \texttt{consE}\ R\ V \mid \texttt{inferI} \mid \texttt{inferE}\ R\ V \\
\\
\text{Inferables} & R & ::= & \pi \mid \texttt{check}\ V\ s\ u_1\ u_2 \mid \texttt{conjE1}\ R \mid \texttt{conjE2}\ R \mid \texttt{impE}\ R\ V\ u_1\ u_2 \\
& & & \mid \texttt{forallE}\ t\ R
\end{array}
$$

Judgments and Inference Rules. We describe the use of proof terms through a proof term calculus, which contains two hypothetical judgments $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ (checking) and $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ (inference). These judgments are are defined by the mutually inductive rules in Figures 5.1 and 5.2. The hypotheses Π in these figures

are a multiset $\pi_1 : J_1, \ldots, \pi_n : J_n$ of pairs of proof variables π_i and judgments J_i. Each hypothesis J_i may be referred to in proof terms using the name π_i. (We assume implicitly that the names $\pi_1, \ldots, \pi_n$ are distinct.)

$$\Pi \quad ::= \quad \pi_1 : J_1, \ldots, \pi_n : J_n$$

The definition of the restriction operator $\cdot|$ is revised to retain names of assumptions:

$$\Pi| = \{(\pi : k \ \mathsf{claims} \ r \circ [u_1, u_2]) \in \Pi\}$$

The rules defining the judgments $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ are similar to those for canonical and atomic proofs from §4.5 – wherever a canonical proof judgment $\Uparrow$ appeared in the rules of §4.5, we now have a checking judgment $\Longleftarrow$, and wherever an atomic proof judgment $\Downarrow$ appeared in the rules of §4.5 we now have an inference judgment $\Longrightarrow$. There is, of course, the rule (check) which has no analogue in the rules of §4.5, and it is this rule that allows proof terms to witness all natural deduction proofs, not just canonical and atomic ones. With the exception of the last one, the following observations about the proof term calculus of Figures 5.1 and 5.2 parallel similar observations from §4.5.

1. The conclusion of every introduction rule has a checkable proof term, and so does its principal premise.
2. The conclusion of every let-like elimination rule has a checkable proof term.
3. The conclusions of rules (check), (hyp), (claims), ($\wedge$E$_1$), ($\wedge$E$_2$), ($\supset$E), and ($\forall$E) contain inferable proof terms.
4. The principal premise of every elimination rule has an inferable proof term.
5. The rule (infer) shifts judgments from inference to checking, whereas the rule (check) shifts in the other direction.

While a precise connection between the proof term calculus of Figures 5.1 and 5.2 and canonical and atomic proofs is postponed to §5.4, in the following we explain how proof terms may be used to witness natural deduction proofs in general.

5.1.1 Connection to Natural Deduction

The proof term calculus of Figures 5.1 and 5.2 corresponds to natural deduction of Figures 4.2 and 4.3 due to two properties that we formalize and prove in this section. First, if we erase the proof term, all proof variables, and the entailment symbol $\Longrightarrow$ or $\Longleftarrow$ from a hypothetical judgment that is provable in the proof term calculus, then we obtain a hypothetical judgment that is provable in natural deduction. Thus the proof term calculus is *sound* with respect to natural deduction. Second, given a natural deduction proof of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and any Π that assigns unique names to judgments in Γ, we

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1,u_2] \qquad \Sigma;\Psi \models u_1 \leq u_1' \qquad \Sigma;\Psi \models u_2' \leq u_2}{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longleftarrow s \circ [u_1',u_2']}\text{infer}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{check}\ V\ s\ u_1\ u_2 \Longrightarrow s \circ [u_1,u_2]}\text{check}$$

$$\frac{}{\Sigma;\Psi;E;\Pi,\pi : s \circ [u_1,u_2] \vdash^{\nu} \pi \Longrightarrow s \circ [u_1,u_2]}\text{hyp}$$

$$\frac{\nu = k,u_b,u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2 \qquad \Sigma;\Psi \models k' \succeq k}{\Sigma;\Psi;E;\Pi,\pi : k' \ \mathsf{claims}\ s \circ [u_1,u_2] \vdash^{\nu} \pi \Longrightarrow s \circ [u_1,u_2]}\text{claims}$$

$$\frac{\Sigma;\Psi;E;\Pi| \vdash^{k,u_1,u_2} V \Longleftarrow s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{saysI}\ V \Longleftarrow k\ \mathsf{says}\ s \circ [u_1,u_2]}\mathsf{saysI}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longrightarrow k\ \mathsf{says}\ s \circ [u_1,u_2] \qquad \Sigma;\Psi;E;\Pi,\pi : k\ \mathsf{claims}\ s \circ [u_1,u_2] \vdash^{\nu} V \Longleftarrow s' \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{saysE}\ R\ (\pi.V) \Longleftarrow s' \circ [u_1',u_2']}\mathsf{saysE}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{atI}\ V \Longleftarrow (s\ @\ [u_1,u_2]) \circ [u_1',u_2']}@\text{I}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longrightarrow s\ @\ [u_1,u_2] \circ [u_1',u_2'] \qquad \Sigma;\Psi;E;\Pi,\pi : s \circ [u_1,u_2] \vdash^{\nu} V \Longleftarrow s' \circ [u_1'',u_2'']}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{atE}\ R\ (\pi.V) \Longleftarrow s' \circ [u_1'',u_2'']}@\text{E}$$

$$\frac{\Sigma;\Psi \models c}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{consI} \Longleftarrow c \circ [u_1,u_2]}\text{consI}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longrightarrow c \circ [u_1,u_2] \qquad \Sigma;\Psi,c;E;\Pi \vdash^{\nu} V \Longleftarrow s' \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{consE}\ R\ V \Longleftarrow s' \circ [u_1',u_2']}\text{consE}$$

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{interI} \Longleftarrow i \circ [u_1,u_2]}\text{interI}$$

$$\frac{\Sigma;\Psi;E;\Pi \vdash^{\nu} R \Longrightarrow i \circ [u_1,u_2] \qquad \Sigma;\Psi;E,i;\Pi \vdash^{\nu} V \Longleftarrow s' \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi \vdash^{\nu} \texttt{interE}\ R\ V \Longleftarrow s' \circ [u_1',u_2']}\text{interE}$$

Figure 5.1: Bidirectional proof terms, part 1

can construct a checkable V and a derivation in the proof term calculus which shows that $\Sigma;\Psi;E;\Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1,u_2]$. This implies that the proof term calculus is *complete* with respect to natural deduction.

Definition 5.1. For $\Pi = \pi_1 : J_1,\ldots,\pi_n : J_n$, define $|\Pi|$ to be hypotheses $J_1,\ldots,J_n$.

Theorem 5.2 (Soundness)**.** *The following hold.*

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} s_1\circ V_1 \Longleftarrow [u_1,u_2] \qquad \Sigma;\Psi;E;\Pi\vdash^{\nu} V_2 \Longleftarrow s_2\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjI}\ V_1\ V_2 \Longleftarrow s_1\wedge s_2\circ[u_1,u_2]}\wedge\mathrm{I}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\wedge s_2\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjE1}\ R \Longrightarrow s_1\circ[u_1,u_2]}\wedge\mathrm{E}_1 \qquad \frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\wedge s_2\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjE2}\ R \Longrightarrow s_2\circ[u_1,u_2]}\wedge\mathrm{E}_2$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_1\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjI1}\ V \Longleftarrow s_1\vee s_2\circ[u_1,u_2]}\vee\mathrm{I}_1$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_2\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjI2}\ V \Longleftarrow s_1\vee s_2\circ[u_1,u_2]}\vee\mathrm{I}_2$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\vee s_2\circ[u_1,u_2]\\ \Sigma;\Psi;E;\Pi,\pi_1 : s_1\circ[u_1,u_2]\vdash^{\nu} V_1 \Longleftarrow s'\circ[u_1',u_2']\\ \Sigma;\Psi;E;\Pi,\pi_2 : s_2\circ[u_1,u_2]\vdash^{\nu} V_2 \Longleftarrow s'\circ[u_1',u_2']\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjE}\ R\ (\pi_1.V_1)\ (\pi_2.V_2) \Longleftarrow s'\circ[u_1',u_2']}\vee\mathrm{E}$$

$$\frac{}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{topI} \Longleftarrow \top\circ[u_1,u_2]}\top\mathrm{I} \qquad \frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow \bot\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{botE}\ R \Longleftarrow s\circ[u_1',u_2']}\bot\mathrm{E}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time};\Psi, u_1\le x_1, x_2\le u_2;E;\Pi,\pi : s_1\circ[x_1,x_2]\vdash^{\nu} V \Longleftarrow s_2\circ[x_1,x_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1\supset s_2\circ[u_1,u_2]}\supset\mathrm{I}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\supset s_2\circ[u_1,u_2]\\ \Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_1\circ[u_1',u_2'] \qquad \Sigma;\Psi\models u_1\le u_1' \qquad \Sigma;\Psi\models u_2'\le u_2\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{impE}\ R\ V\ u_1'\ u_2' \Longrightarrow s_2\circ[u_1',u_2']}\supset\mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s\circ[u_1,u_2]}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{forallI}\ (x.V) : \forall x{\Longleftarrow}\sigma.s\circ[u_1,u_2]}\forall\mathrm{I}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R : \forall x{\Longrightarrow}\sigma.s\circ[u_1,u_2] \qquad \Sigma\vdash t:\sigma}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{forallE}\ t\ R \Longrightarrow s[t/x]\circ[u_1,u_2]}\forall\mathrm{E}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s[t/x]\circ[u_1,u_2] \qquad \Sigma\vdash t:\sigma}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{existsI}\ t\ V : \exists x{\Longleftarrow}\sigma.s\circ[u_1,u_2]}\exists\mathrm{I}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow \exists x{:}\sigma.s\circ[u_1,u_2] \qquad \Sigma, x{:}\sigma;\Psi;E;\Pi,\pi : s\circ[u_1,u_2]\vdash^{\nu} V \Longleftarrow s'\circ[u_1',u_2']}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{existsI}\ (x.\pi.V) \Longleftarrow s'\circ[u_1',u_2']}\exists\mathrm{E}$$

Figure 5.2: Bidirectional proof terms, part 2

1. *If* $\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s\circ[u_1,u_2]$, *then* $\Sigma;\Psi;E;|\Pi|\vdash^{\nu} s\circ[u_1,u_2]$.

2. *If* $\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s\circ[u_1,u_2]$, *then* $\Sigma;\Psi;E;|\Pi|\vdash^{\nu} s\circ[u_1,u_2]$.

Proof. By simultaneous induction on given derivations and case analysis of their last rules. For the cases of rules (hyp) and (claims) of Figure 5.1 we use the property (C-refl-time) from §4.2.1. The case of rule (infer) relies on Theorem 4.4. □

An important point about this soundness result is that its proof is not based in simply erasing all proof variables and proof terms from the given proof term calculus derivations. Indeed, doing the latter may not result in a valid natural deduction derivation because after erasing proof terms, the rule (infer) in the proof term calculus corresponds exactly to time subsumption for natural deduction, which is not an explicit rule but established as a theorem (Theorem 4.4).

Theorem 5.3 (Completeness)**.** *Suppose* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *and* $|\Pi| = \Gamma$*. Then there is a checkable* V *such that* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and a case analysis of its last rule. Some interesting cases are shown below.

Case. $\dfrac{\Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\text{hyp}$

Let $|\Pi, \pi : s \circ [u_1', u_2']| = \Gamma, s \circ [u_1', u_2']$. Choose $V = \pi$.
To show: $\Sigma; \Psi; E; \Pi, \pi : s \circ [u_1', u_2'] \vdash^{\nu} \pi \Longleftarrow s \circ [u_1, u_2]$.

1. $\Sigma; \Psi; E; \Pi, \pi : s \circ [u_1', u_2'] \vdash^{\nu} \pi \Longrightarrow s \circ [u_1', u_2']$ (Rule (hyp))
2. $\Sigma; \Psi; E; \Pi, \pi : s \circ [u_1', u_2'] \vdash^{\nu} \pi \Longleftarrow s \circ [u_1, u_2]$ (Rule (infer) on 1 and premises)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysI}$

To show: There is a V such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow k \ \mathsf{says}\ s \circ [u_1, u_2]$.

1. $|(\Pi|)| = (|\Pi|)| = \Gamma|$ (Definition)
2. There is a V' such that $\Sigma; \Psi; E; \Pi| \vdash^{k, u_1, u_2} V' \Longleftarrow s \circ [u_1, u_2]$ (i.h. on premise; 1)
3. $\Sigma; \Psi; E; \Pi \vdash^{\nu} \texttt{saysI}\ V' \Longleftarrow k \ \mathsf{says}\ s \circ [u_1, u_2]$ (Rule (saysI) on 2)

Choose $V = \texttt{saysI}\ V'$.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1', u_2'] \qquad \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1', u_2']}\supset\text{E}$

To show: There is a V such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s_2 \circ [u_1', u_2']$

1. There is a V_1 such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V_1 \Longleftarrow s_1 \supset s_2 \circ [u_1, u_2]$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Pi \vdash^{\nu} \texttt{check}\ V_1\ (s_1 \supset s_2)\ u_1\ u_2 \Longrightarrow s_1 \supset s_2 \circ [u_1, u_2]$ (Rule (check) on 1)

3. There is a V_2 such that $\Sigma; \Psi; E; \Gamma \vdash^{\nu} V_2 \Longleftarrow s_1 \circ [u_1', u_2']$ (i.h. on 2nd premise)
4. $\Sigma; \Psi; E; \Pi \vdash^{\nu} \texttt{impE}\ (\texttt{check}\ V_1\ (s_1 \supset s_2)\ u_1\ u_2)\ V_2\ u_1'\ u_2' \Longrightarrow s_2 \circ [u_1', u_2']$
(Rule ($\supset$E) on 2,3, and 3rd,4th premises)
5. $\Sigma; \Psi \models u_1' \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2'$ ((C-refl-time) from §4.4)
6. $\Sigma; \Psi; E; \Pi \vdash^{\nu} \texttt{impE}\ (\texttt{check}\ V_1\ (s_1 \supset s_2)\ u_1\ u_2)\ V_2\ u_1'\ u_2' \Longleftarrow s_2 \circ [u_1', u_2']$
(Rule (infer) on 4,5)

Choose $V = \texttt{impE}\ (\texttt{check}\ V_1\ (s_1 \supset s_2)\ u_1\ u_2)\ V_2\ u_1'\ u_2'$. □

The proof of Theorem 5.3 is constructive – its inductive cases can be interpreted as a function for constructing the checkable V as well as the derivation of $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ from a proof of $\Sigma; \Psi; E; |\Pi| \vdash^{\nu} s \circ [u_1, u_2]$. It is also instructive to observe the role of the constructors `check` and `infer` in the proof.

5.1.2 Properties of Proof Terms

In general, analogues of all properties of natural deduction from §4.2.3 may be stated and proved for the proof term calculus. This section presents some of the more interesting properties, including the analogue of the time subsumption principle. As for natural deduction, the following view subsumption principle is needed to prove time subsumption.

Theorem 5.4 (View subsumption)**.** *Suppose the following hold:*

1. $\nu = k_0, u_b, u_e$ *and* $\nu' = k_0', u_b', u_e'$
2. $\Sigma; \Psi \models k_0 \succeq k_0'$, $\Sigma; \Psi \models u_b \leq u_b'$, *and* $\Sigma; \Psi \models u_e' \leq u_e$.

Then,

A. $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Pi \vdash^{\nu'} V \Longleftarrow s \circ [u_1, u_2]$ *by a derivation of smaller or equal depth.*

B. $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Pi \vdash^{\nu'} R \Longrightarrow s \circ [u_1, u_2]$ *by a derivation of smaller or equal depth.*

Proof. By simultaneous induction on given derivations in A and B, and case analysis of the last rule. The case where the derivation in A ends in (claims) uses the assumptions (C-trans-time) and (C-trans-prin) from §4.2.1, as in the proof of Theorem 4.3. □

The next theorem formally states the time subsumption principle with proof terms. The theorem mentions checkables only since an analogous principle does not hold for inferables, e.g., according to the rule (hyp) of Figure 5.1, the hypothesis $\pi : s \circ [u_1, u_2]$ can be used to infer $s \circ [u_1, u_2]$, but not $s \circ [u_1', u_2']$ for any other interval $[u_1', u_2']$. This restriction in the rule (hyp), and also (claims), is deliberate since it simplifies proof checking. The proof verifier in PCFS expects checkables, not inferables, so having the time subsumption principle only for checkables suffices for our purposes.

Theorem 5.5 (Time subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$ *and* $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_n, u_m]$

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ and case analysis of its last rule. The case where the derivation ends in rule (saysI) uses view subsumption (Theorem 5.4). The case where the derivation ends in ($\supset$I) requires a lemma about substitution of constraints. See Theorem C.2 in Appendix C for details. □

Let $\Xi[\Xi'/\pi]$ denote the usual capture avoiding substitution of Ξ' for variable π in Ξ. The syntax of proof terms is not closed under substitution of proof variables by checkables. For instance, substituting the checkable V for π_1 in the inferable $\pi_1\ \pi_2$ results in $V\ \pi_2$, which is neither a checkable nor an inferable. However, checkables and inferables are individually closed under substitution by inferables, as the following theorem states.

Theorem 5.6 (Closure under substitution)**.** *The following hold.*

1. *For every checkable V and inferable R', $V[R'/\pi]$ is a checkable.*
2. *For every checkable R and inferable R', $R[R'/\pi]$ is an inferable.*

Proof. By simultaneous induction on the structures of V and R. □

With Theorem 5.6 in mind, the following theorem generalizes the substitution principle of natural deduction (Theorem 4.5).

Theorem 5.7 (Substitution)**.** *Suppose* $\Sigma; \Psi; E; \Pi \vdash^{\nu} R' \Longrightarrow s \circ [u_1, u_2]$*. Then the following hold.*

1. $\Sigma; \Psi; E; \Pi, \pi : s \circ [u_1, u_2] \vdash^{\nu} V \Longleftarrow r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V[R'/\pi] \Longleftarrow r \circ [u_1', u_2']$.
2. $\Sigma; \Psi; E; \Pi, \pi : s \circ [u_1, u_2] \vdash^{\nu} R \Longrightarrow r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Pi \vdash^{\nu} R[R'/\pi] \Longrightarrow r \circ [u_1', u_2']$.

Proof. By simultaneous induction on derivations given in (1) and (2), and case analysis of their last rules. □

5.1.3 Bidirectional Verification (The One Not Used in PCFS)

The rules of Figures 5.1 and 5.2 can be interpreted as a decision procedure for verification of proof terms (next theorem). However, for reasons mentioned in the opening of this chapter, the actual verification procedure used in PCFS is different; it is described in §5.2. The following theorem is presented primarily to explain the bidirectional nature of proof verification, and to point out the need for annotations present in the constructors `impE` and `check`.

Theorem 5.8 (Bidirectional verification)**.** *Suppose that the judgments* $\Sigma \vdash t : \sigma$, $\Sigma; \Psi \models c$, *and* $\Sigma; E \models i$ *can all be decided in constant time. Then there are mutually inductive decision procedures* $\mathcal{A}$ *and* $\mathcal{B}$*, with time complexities linear in the sizes of* V *and* R *respectively, such that:*

- $\mathcal{A}$ *takes* $\Sigma, \Psi, E, \Pi, \nu, s, u_1, u_2$, *and* V *as arguments and determines whether* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ *or not.*
- $\mathcal{B}$ *takes* $\Sigma, \Psi, E, \Pi, \nu$, *and* R *as arguments and finds (the unique)* s, u_1, u_2 *such that* $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ *if such* s, u_1, u_2 *exist, else returns an error.*

Proof. By simultaneous construction of $\mathcal{A}$ and $\mathcal{B}$. Each procedure works by a case analysis of the top level constructor of the proof term (V or R) provided as an argument. We show some interesting cases of each procedure. Termination of $\mathcal{A}$ and $\mathcal{B}$ follows by a lexicographic induction, first on the sizes of V and R, and then on the order $\mathcal{A} > \mathcal{B}$.

Procedure $\mathcal{A}$

Case. $V = R$. In this case, the judgment $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$, if provable, can only be established by the rule (infer). Therefore, $\mathcal{A}$ works as follows.

1. Call $\mathcal{B}$ with arguments $\Sigma, \Psi, E, \Pi, \nu$, and R.
2. If (1) returns an error, the required judgment $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ is not provable.
3. If (1) returns s', u_1', u_2', check that $s' = s$, $\Sigma; \Psi \models u_1' \leq u_1$, and $\Sigma; \Psi \models u_2 \leq u_2'$. If all checks succeed then $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ is provable, else it is not.

Procedure $\mathcal{B}$

Case. $R = \texttt{impE}\ R'\ V\ u_1'\ u_2'$. In this case, if s, u_1, u_2 exist, then the judgment $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ can only be established by the rule ($\supset$E). Therefore, $\mathcal{B}$ works as follows.

1. Call $\mathcal{B}$ with arguments $\Sigma, \Psi, E, \Pi, \nu$, and R'.
2. If (1) returns an error, return an error.
3. If (1) returns s', u_1, u_2, check that $s' = s_1 \supset s_2$ for some s_1 and s_2 (if not, return an error).
4. Call $\mathcal{A}$ with arguments $\Sigma, \Psi, E, \Pi, \nu, s_1, u_1', u_2'$ and V.
5. If (4) returns false, return an error.
6. If (4) returns true, check that $\Sigma; \Psi \models u_1 \leq u_1'$ and that $\Sigma; \Psi \models u_2' \leq u_2$. If both checks succeed, return s_2, u_1', u_2' else return an error.

Case. $R = \texttt{check}\ V\ s\ u_1\ u_2$. Following rule (check), $\mathcal{B}$ works as follows.

1. Call $\mathcal{A}$ with arguments $\Sigma, \Psi, E, \Pi, \nu, s, u_1, u_2$, and V.
2. If (1) returns false, return an error.
3. If (1) returns true, return s, u_1, u_2.

□

The reader may observe in the proof the critical roles played by the annotations u_1', u_2' in the case for the inferable $\texttt{impE}\ R\ V\ u_1'\ u_2'$ and the annotations s, u_1, u_2 in the case for the inferable $\texttt{check}\ V\ s\ u_1\ u_2$.

5.2 Proof Verification in PCFS

We mentioned in §4.3.1 that in PCFS an offline proof verifier, called in advance of access, checks proofs of hypothetical judgments of the form $\Sigma; \cdot; E; \Gamma \vdash^{\nu} \texttt{admin says}\ (\texttt{may}\ k\ f\ \eta) \circ [u, u]$. Since proofs in PCFS are represented using checkable proof terms V, the problem of proof verification in PCFS is more precisely one of checking provability of proof term judgments of the form $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \texttt{admin says}\ (\texttt{may}\ k\ f\ \eta) \circ [u, u]$. If such a judgment is provable, the output of proof verification is a signed capability, or procap, that contains the terms k, f, η, else the output is an error. The procap, if obtained, may be used by k to authorize permission η on file f in the PCFS back end (§2). The relevant question here is how the proof verifier determines whether $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \texttt{admin says}\ (\texttt{may}\ k\ f\ \eta) \circ [u, u]$ is provable or not. Whereas a casual inspection of the problem may suggest that the procedures constructed in Theorem 5.8 would suffice for this purpose, this is not actually the case in PCFS because u and E are not known when the proof verifier is invoked. Recall from §4.3.1 that u and E represent respectively the exact time at which the access authorized by the proof is *used* and the system state prevalent at that time. It is clearly impractical to expect either the verifier or the user invoking the verifier to determine either of these in advance of access. In the absence of knowledge of u and E, the procedures of Theorem 5.8 cannot be used directly.

As a result the proof verifier in PCFS uses a modified verification procedure that postpones checking of constraints depending on u as well as interpreted predicates depending on E to the back end. This is done by writing such constraints and interpreted predicates to the procap, whose validity becomes *conditional* on the satisfaction of the constraints and the interpreted predicates. More precisely, instead of ascertaining provability of the judgment $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \texttt{admin says}\ (\texttt{may}\ k\ f\ \eta) \circ [u, u]$, the proof verifier ascertains the provability of the following judgment, which replaces u by a symbolic constant ctime that the back end recognizes, and replaces E by the empty system state $\cdot$.

$$\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \texttt{admin says}\ (\texttt{may}\ k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}]$$

In ascertaining the provability of this judgment, the proof verifier may encounter several judgments of the form $\Sigma'; \Psi' \models c'$ containing the symbolic constant ctime, which the decision procedure for constraints is unable to discharge. All such judgments are written to the

procap as conditions. (Note that Ψ' may not always be empty as hypothetical constraints may be introduced in some branches of the proof by rules like (consE).) Similarly, the proof verifier may encounter judgments of the form $\Sigma'; E' \models i'$ which cannot be verified because E' does not include the intended state E. Again all such judgments are written to the procap as conditions.

The back end of PCFS "completes" the proof verification whenever it uses a procap for authorizing access by making the following two checks.

- For each judgment $\Sigma', \mathsf{ctime{:}time}; \Psi' \models c'$ written as a condition in the procap, it checks that $\Sigma'; \Psi'[u/\mathsf{ctime}] \models c'[u/\mathsf{ctime}]$, where u is the clock reading at the time of the access.

- For each judgment $\Sigma', \mathsf{ctime{:}time}; E' \models i'$ written as a condition in the procap, it checks that $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$, where E is the state prevailing at the time of the access.

It is not immediately obvious that this two-part proof verification, where some checks are performed prior to access by an offline verifier and the remaining during a file system call by the back end, is actually correct, i.e. it authorizes exactly those accesses that a proof verifier with full knowledge of u and E would authorize using the procedures constructed in Theorem 5.8. The latter would be the case if we were to follow the usual proof-carrying authorization approach and embed the proof verifier in the PCFS back end. In the rest of this section, we formalize the procedure that the PCFS proof verifier uses to ascertain the provability of $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \texttt{admin says } (\texttt{may } k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}]$ and to generate conditions from it, and show that the two-part proof verification that PCFS employs is correct in the sense mentioned above.

5.2.1 The PCFS Proof Verifier

In this section we formally describe the PCFS proof verifier that generates conditions for procaps; the next section shows that the checks it makes together with the checks made by the back end ensure correct proof verification. We use the term *hypothetical constraint* for judgments of the form $\Sigma; \Psi \models c$ (the judgment may or may not hold). Similarly, we call judgments of the form $\Sigma; E \models i$ *hypothetical states*. The symbols $\mathcal{C}$ and $\mathcal{I}$ denote multisets of hypothetical constraints and hypothetical states respectively. We define two functions, both named `unsat`, that take multisets of hypothetical constraints and multisets of hypothetical states as arguments and return the subsets that do not hold.

$$\begin{aligned} \texttt{unsat}(\mathcal{C}) &= \{(\Sigma; \Psi \models c) \in \mathcal{C} \mid \Sigma; \Psi \models c \text{ does not hold}\} \\ \texttt{unsat}(\mathcal{I}) &= \{(\Sigma; E \models i) \in \mathcal{I} \mid \Sigma; E \models i \text{ does not hold}\} \end{aligned}$$

The proof verification procedure used in PCFS is described by two judgments:

$$\begin{aligned} &\text{(Checking)} \quad \Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I} \\ &\text{(Inference)} \quad \Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I} \end{aligned}$$

These judgments are respective generalizations of the checking and synthesis judgments of §5.1, with the added proviso that there are outputs $\mathcal{C}$ and $\mathcal{I}$ associated with each judgment. Intuitively,

- $\mathcal{C}$ contains exactly those hypothetical constraints that are needed for the given proof to be correct, but cannot be shown to hold because they contain uninstantiated constants like ctime.

- $\mathcal{I}$ contains exactly those hypothetical states that are needed for the given proof to be correct, but cannot be shown to hold because the system state assumed in them is incomplete.

The inference rules defining the two judgments are listed in Figures 5.3 and 5.4. These rules are in one-to-one correspondence with those of the proof term calculus and they have been obtained by applying the above two principles to the rules of Figures 5.1 and 5.2. The main idea is that whenever a hypothetical constraint or hypothetical state that does not hold is encountered, it is written to the output $\mathcal{C}$ or $\mathcal{I}$. (Observe the use of the functions `unsat` in the rules.) The rules may be interpreted as decision procedures in the following sense.

Theorem 5.9. *Assuming that the judgments $\Sigma \vdash t : \sigma$, $\Sigma; \Psi \models c$, and $\Sigma; E \models i$ are all decidable, there are mutually inductive decision procedures $\mathcal{A}$ and $\mathcal{B}$ such that:*

- *$\mathcal{A}$ takes $\Sigma, \Psi, E, \Pi, \nu, s, u_1, u_2$, and V as arguments and finds (the unique) $\mathcal{C}$ and $\mathcal{I}$ such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ if such $\mathcal{C}$ and $\mathcal{I}$ exist, else returns an error.*

- *$\mathcal{B}$ takes $\Sigma, \Psi, E, \Pi, \nu$, and R as arguments and finds (the unique) $s, u_1, u_2, \mathcal{C}, \mathcal{I}$ such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ if such $s, u_1, u_2, \mathcal{C}, \mathcal{I}$ exist, else returns an error.*

Further, if $\Sigma \vdash t : \sigma$, $\Sigma; \Psi \models c$, and $\Sigma; E \models i$ can be decided in constant time, then the running times of $\mathcal{A}$ and $\mathcal{B}$ can be made linear in the sizes of V and R respectively.

Proof. By construction of $\mathcal{A}$ and $\mathcal{B}$, as in the proof of Theorem 5.8. □

The PCFS proof verification tool uses exactly procedure $\mathcal{A}$ of Theorem 5.9 to establish a judgment of the form $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \mathtt{admin}\ \mathsf{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$ (see details below). If the judgment can be established, then $\mathcal{C}$ and $\mathcal{I}$ become conditions of the procap generated by the proof verifier. These conditions are checked by the back end of PCFS. Therefore, most of the proof checking is performed in the PCFS front end by the tool called the "proof verifier" whereas the part of the proof that depends on the time of access or system state is checked in the back end during file access. The entire process of proof verification executed in the two parts is summarized below.

1. In the front end:

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s\circ[u_1,u_2]\searrow\mathcal{C}';\mathcal{I} \qquad \mathcal{C}=(\Sigma;\Psi\models u_1\leq u_1'),(\Sigma;\Psi\models u_2'\leq u_2)}{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longleftarrow s\circ[u_1',u_2']\searrow\mathcal{C}',\mathtt{unsat}(\mathcal{C});\mathcal{I}}\text{infer}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s\circ[u_1,u_2]\searrow\mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \mathtt{check}\ V\ s\ u_1\ u_2 \Longrightarrow s\circ[u_1,u_2]\searrow\mathcal{C};\mathcal{I}}\text{check}$$

$$\frac{}{\Sigma;\Psi;E;\Pi,\pi:s\circ[u_1,u_2]\vdash^{\nu}\pi\Longrightarrow s\circ[u_1,u_2]\searrow\cdot;\cdot}\text{hyp}$$

$$\frac{\nu=k,u_b,u_e \qquad \mathcal{C}=(\Sigma;\Psi\models u_1\leq u_b),(\Sigma;\Psi\models u_e\leq u_2),(\Sigma;\Psi\models k'\succeq k)}{\Sigma;\Psi;E;\Pi,\pi:k'\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu}\pi\Longrightarrow s\circ[u_1,u_2]\searrow\mathtt{unsat}(\mathcal{C});\cdot}\text{claims}$$

$$\frac{\Sigma;\Psi;E;\Pi|\vdash^{k,u_1,u_2} V\Longleftarrow s\circ[u_1,u_2]\searrow\mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{saysI}\ V\Longleftarrow k\ \mathsf{says}\ s\circ[u_1,u_2]\searrow\mathcal{C};\mathcal{I}}\mathsf{saysI}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R\Longrightarrow k\ \mathsf{says}\ s\circ[u_1,u_2]\searrow\mathcal{C}_1;\mathcal{I}_1\\ \Sigma;\Psi;E;\Pi,\pi:k\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu} V\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_2;\mathcal{I}_2\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{saysE}\ R\ (\pi.V)\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}\mathsf{saysE}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V\Longleftarrow s\circ[u_1,u_2]\searrow\mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{atI}\ V\Longleftarrow(s\ @\ [u_1,u_2])\circ[u_1',u_2']\searrow\mathcal{C};\mathcal{I}}@\text{I}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R\Longrightarrow s\ @\ [u_1,u_2]\circ[u_1',u_2']\searrow\mathcal{C}_1;\mathcal{I}_1\\ \Sigma;\Psi;E;\Pi,\pi:s\circ[u_1,u_2]\vdash^{\nu} V\Longleftarrow s'\circ[u_1'',u_2'']\searrow\mathcal{C}_2;\mathcal{I}_2\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{atE}\ R\ (\pi.V)\Longleftarrow s'\circ[u_1'',u_2'']\searrow\mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}@\text{E}$$

$$\frac{\mathcal{C}=(\Sigma;\Psi\models c)}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{consI}\Longleftarrow c\circ[u_1,u_2]\searrow\mathtt{unsat}(\mathcal{C});\cdot}\text{consI}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R\Longrightarrow c\circ[u_1,u_2]\searrow\mathcal{C}_1;\mathcal{I}_1 \qquad \Sigma;\Psi,c;E;\Pi\vdash^{\nu} V\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_2;\mathcal{I}_2}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{consE}\ R\ V\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}\text{consE}$$

$$\frac{\mathcal{I}=(\Sigma;E\models i)}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{interI}\Longleftarrow i\circ[u_1,u_2]\searrow\cdot;\mathtt{unsat}(\mathcal{I})}\text{interI}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R\Longrightarrow i\circ[u_1,u_2]\searrow\mathcal{C}_1;\mathcal{I}_1 \qquad \Sigma;\Psi;E,i;\Pi\vdash^{\nu} V\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_2;\mathcal{I}_2}{\Sigma;\Psi;E;\Pi\vdash^{\nu}\mathtt{interE}\ R\ V\Longleftarrow s'\circ[u_1',u_2']\searrow\mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}\text{interE}$$

Figure 5.3: PCFS proof verification, part 1

(a) A user invokes the PCFS proof verifier and provides to it Π (in the form of certificates), V (which the user has constructed), k, f, and η.

(b) The proof verifier reads Σ and admin from a protected location (see §7 for details).

(c) Using procedure $\mathcal{A}$ of Theorem 5.9, the verifier tries to construct $\mathcal{C}$ and $\mathcal{I}$ such

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} s_1\circ V_1 \Longleftarrow [u_1,u_2]\searrow \mathcal{C}_1;\mathcal{I}_1 \qquad \Sigma;\Psi;E;\Pi\vdash^{\nu} V_2 \Longleftarrow s_2\circ [u_1,u_2]\searrow \mathcal{C}_2;\mathcal{I}_2}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjI}\ V_1\ V_2 \Longleftarrow s_1\wedge s_2\circ [u_1,u_2]\searrow \mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}\wedge\mathrm{I}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\wedge s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjE1}\ R \Longrightarrow s_1\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\wedge\mathrm{E}_1$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\wedge s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{conjE2}\ R \Longrightarrow s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\wedge\mathrm{E}_2$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_1\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjI1}\ V \Longleftarrow s_1\vee s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\vee\mathrm{I}_1$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjI2}\ V \Longleftarrow s_1\vee s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\vee\mathrm{I}_2$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\vee s_2\circ [u_1,u_2]\searrow \mathcal{C}_1;\mathcal{I}_1\\ \Sigma;\Psi;E;\Pi,\pi_1 : s_1\circ [u_1,u_2]\vdash^{\nu} V_1 \Longleftarrow s'\circ [u_1',u_2']\searrow \mathcal{C}_2;\mathcal{I}_2\\ \Sigma;\Psi;E;\Pi,\pi_2 : s_2\circ [u_1,u_2]\vdash^{\nu} V_2 \Longleftarrow s'\circ [u_1',u_2']\searrow \mathcal{C}_3;\mathcal{I}_3\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{disjE}\ R\ (\pi_1.V_1)\ (\pi_2.V_2) \Longleftarrow s'\circ [u_1',u_2']\searrow \mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3;\mathcal{I}_1,\mathcal{I}_2,\mathcal{I}_3}\vee\mathrm{E}$$

$$\frac{}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{topI} \Longleftarrow \top\circ [u_1,u_2]\searrow \cdot;\cdot}\top\mathrm{I} \qquad \frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow \bot\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{botE}\ R \Longleftarrow s\circ [u_1',u_2']\searrow \mathcal{C};\mathcal{I}}\bot\mathrm{E}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time};\Psi, u_1\le x_1, x_2\le u_2;E;\Pi,\pi : s_1\circ [x_1,x_2]\vdash^{\nu} V \Longleftarrow s_2\circ [x_1,x_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1\supset s_2\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\supset\mathrm{I}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow s_1\supset s_2\circ [u_1,u_2]\searrow \mathcal{C}_1;\mathcal{I}_1\\ \Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s_1\circ [u_1',u_2']\searrow \mathcal{C}_2;\mathcal{I}_2 \qquad \mathcal{C} = (\Sigma;\Psi\models u_1\le u_1'),(\Sigma;\Psi\models u_2'\le u_2)\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{impE}\ R\ V\ u_1'\ u_2' \Longrightarrow s_2\circ [u_1',u_2']\searrow \mathcal{C}_1,\mathcal{C}_2,\mathtt{unsat}(\mathcal{C});\mathcal{I}_1,\mathcal{I}_2}\supset\mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{forallI}\ (x.V) : \forall x{\Longleftarrow}\sigma.s\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\forall\mathrm{I}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow \forall x{:}\sigma.s\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I} \qquad \Sigma\vdash t:\sigma}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{forallE}\ t\ R \Longrightarrow s[t/x]\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\forall\mathrm{E}$$

$$\frac{\Sigma;\Psi;E;\Pi\vdash^{\nu} V \Longleftarrow s[t/x]\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I} \qquad \Sigma\vdash t:\sigma}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{existsI}\ t\ V \Longleftarrow \exists x{:}\sigma.s\circ [u_1,u_2]\searrow \mathcal{C};\mathcal{I}}\exists\mathrm{I}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi\vdash^{\nu} R \Longrightarrow \exists x{:}\sigma.s\circ [u_1,u_2]\searrow \mathcal{C}_1;\mathcal{I}_1\\ \Sigma, x{:}\sigma;\Psi;E;\Pi,\pi : s\circ [u_1,u_2]\vdash^{\nu} V \Longleftarrow s'\circ [u_1',u_2']\searrow \mathcal{C}_2;\mathcal{I}_2\end{array}}{\Sigma;\Psi;E;\Pi\vdash^{\nu} \texttt{existsI}\ (x.\pi.V) \Longleftarrow s'\circ [u_1',u_2']\searrow \mathcal{C}_1,\mathcal{C}_2;\mathcal{I}_1,\mathcal{I}_2}\exists\mathrm{E}$$

Figure 5.4: PCFS proof verification, part 2

that $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \mathtt{admin}\ \mathtt{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$, where ctime is a distinguished constant that the back end recognizes and ν is a view made of three fresh constants.

(d) If (c) returns an error, then verification fails.

(e) If (c) produces $\mathcal{C}$ and $\mathcal{I}$, then the verifier issues to the user a procap that contains the access right $\langle k, f, \eta \rangle$ as well as the conditions $\mathcal{C}$ and $\mathcal{I}$ and signs it with a secret key known only to it and the PCFS back end. The procap is placed in the procap store by the user (see §2).

2. In the back end:

(a) During an access on file f by principal k that requires permission η, the PCFS back end reads the procap generated in step 1(e) from the procap store, and parses out $\mathcal{C}$ and $\mathcal{I}$.

(b) For each hypothetical constraint $\Sigma', \mathsf{ctime{:}time}; \Psi' \models c'$ in $\mathcal{C}$, the PCFS back end checks that $\Sigma'; \Psi'[u/\mathsf{ctime}] \models c'[u/\mathsf{ctime}]$ holds, where u is the time of access.

(c) For each hypothetical state $\Sigma', \mathsf{ctime{:}time}; E' \models i'$ in $\mathcal{I}$, the PCFS back end checks that $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$ where E is the system state prevailing at time u.

(d) If all checks in 2(b) and 2(c) succeed, then access is allowed, else it is denied.

5.2.2 Correctness of PCFS Proof Verification

What we seek to establish now is that the PCFS proof verification procedure summarized at the end of §5.2.1 is both sound and complete. We state soundness and completeness of proof verification in PCFS with respect to particular proofs that the user provides. For soundness we wish to show that successful execution of step 1(c) with a proof term V, followed by successful execution of steps 2(b) and 2(c) on the procap derived from it implies that $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \mathtt{admin}\ \mathtt{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [u, u]$. Dually, completeness means that if $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \mathtt{admin}\ \mathtt{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [u, u]$, then given Π and V as inputs, the proof verifier will successfully execute to produce conditions $\mathcal{C}$ and $\mathcal{I}$, which will then successfully check in the PCFS back end at time u in system state E according to checks 2(b) and 2(c). Soundness follows immediately from the following lemma about the inference system of Figures 5.3 and 5.4.

Lemma 5.10 (Soundness)**.** *Suppose that the following hold for some $\mathcal{C}$, $\mathcal{I}$, list $\vec{x}$ of term variables, list $\vec{\sigma}$ of sorts, list $\vec{t_0}$ of terms satisfying $\Sigma \vdash \vec{t_0} : \vec{\sigma}$, and system state E_0 not containing any element of $\vec{x}$.*

1. *For each $(\Sigma', \vec{x} : \vec{\sigma}; \Psi' \models c') \in \mathcal{C}$, it is the case that $\Sigma'; \Psi'[\vec{t_0}/\vec{x}] \models c'[\vec{t_0}/\vec{x}]$.*

2. *For each $(\Sigma', \vec{x} : \vec{\sigma}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E_0, E'[\vec{t_0}/\vec{x}] \models i'[\vec{t_0}/\vec{x}]$.*

Then,

A. $\Sigma, \vec{x} : \vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *implies* $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V[\vec{t_0}/\vec{x}] \Longleftarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

B. $\Sigma, \vec{x} : \vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *implies* $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longrightarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

Proof. By simultaneous induction on the derivations given in A and B and case analysis of their last rules. See Appendix C, Lemma C.3 for some representative cases. □

Theorem 5.11 (Soundness of PCFS verification)**.** *Suppose that the following hold for some time point u and some ground system state E.*

1. $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow s \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$ *for a fresh constant* ctime *that does not occur in Σ, Π, V, and s.*

2. *For each hypothetical constraint* $(\Sigma', \mathsf{ctime{:}time}; \Psi' \models c') \in \mathcal{C}$ *it is the case that* $\Sigma'; \Psi'[u/\mathsf{ctime}] \models c'[u/\mathsf{ctime}]$.

3. *For each* $(\Sigma', \mathsf{ctime{:}time}; E' \models i') \in \mathcal{I}$ *it is the case that* $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$.

4. $\Sigma \vdash u : \mathsf{time}$.

Then $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u, u]$.

Proof. This theorem is a specific instance of Lemma 5.10(A). □

By choosing u to be the time of access, E to be the system state prevalent at time u, and s to be admin says (may k f η), the assumptions (1)–(3) in the statement of the theorem correspond exactly to the steps 1(c), 2(b), and 2(c) in the summary at the end of §5.2.1, and therefore, this theorem is indeed a statement of the soundness of the PCFS two-part verification procedure.

Dual to soundness, the next theorem shows that the PCFS verification method is complete in the sense described at the beginning of this section.

Theorem 5.12 (Completeness of PCFS verification)**.** *Suppose that* $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u, u]$. *Let* ctime *be a fresh constant. Then there exist $\mathcal{C}$ and $\mathcal{I}$ such that the following hold.*

1. $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow s \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$.

2. *For each* $(\Sigma', \mathsf{ctime{:}time}; \Psi' \models c') \in \mathcal{C}$, *it is the case that* $\Sigma'; \Psi'[u/\mathsf{ctime}] \models c'[u/\mathsf{ctime}]$.

3. *For each* $(\Sigma', \mathsf{ctime{:}time}; E' \models i') \in \mathcal{I}$, *it is the case that* $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$.

Proof. See Theorem C.5 in Appendix C. The proof is based on a converse of Lemma 5.10, which is also presented in Appendix C. □

Clause (1) of this theorem states that the offline proof verifier of PCFS will always succeed in checking the structure of a proof that is valid at some time u in some system state E. Clauses (2) and (3) imply that the conditions generated by the verifier will successfully check in the back end at time u in system state E to allow access. A practically useful consequence follows from a combination of this theorem with time subsumption (Theorem 5.5): If $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ for some time interval $[u_1, u_2]$, then the conditions $\mathcal{C}$ and $\mathcal{I}$ resulting from the verification of V can be successfully checked at *any* time point u in the interval $[u_1, u_2]$ (provided that the state E holds at u). As a result, the proof verifier needs to be invoked only once for each proof term and the procap generated from it can be used again and again over the entire interval of time for which the proof term authorizes access.

5.2.3 Procaps

A procap, is a cryptographic token issued by the proof verifier after it successfully checks a proof. It allows a single user a specific permission on one file or directory. Formally, a procap is a six-tuple $\langle k, f, \eta, \mathcal{C}, \mathcal{I}, sig \rangle$ where

- k is the principal who is authorized access by the procap.
- f is the file or directory to which access is authorized.
- η is the permission allowed (PCFS uses five permissions – read, write, execute, identity, and govern).
- $\mathcal{C}$ is the multiset of hypothetical constraints on which the procap is conditional; they may contain the symbolic constant ctime.
- $\mathcal{I}$ is the multiset of hypothetical states on which the procap is conditional.
- sig is a cryptographic signature over the first five elements of the tuple, created using a symmetric key known only to the proof verifier and the PCFS back end. The signature prevents forging of procaps.

The procap $\langle k, f, \eta, \mathcal{C}, \mathcal{I}, sig \rangle$ is issued by the proof verifier in the front end whenever it successfully checks that the judgment $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \mathtt{admin}\ \mathsf{says}\ (\mathtt{may}\ k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$ can be established. In order to use a procap, the back end must verify its cryptographic signature sig, and check its conditions $\mathcal{C}$ and $\mathcal{I}$ according to steps 2(b) and 2(c) listed at the end of §5.2.1.

Procaps are central in PCFS since they carry information about access rights $\langle k, f, \eta \rangle$, as well as conditions $\mathcal{C}$ and $\mathcal{I}$ on which the rights are contingent, from the proof verifier in the front end to the reference monitor in the back end. In fact the structure of procaps is the only entity in PCFS that the front end and back end must agree on. Other than that the two parts of PCFS are totally independent. The set of all valid procaps in a PCFS system can also be considered a cache of proofs that have been verified, even though procaps differ from traditional caches in that procaps can be individually distributed and they are conditional on system state and time.

5.2.4 Revocation of Policy Rules

So far we have assumed implicitly that any policy rules (Π) in effect at the time of proof verification in the front end are also in effect at the time of access. This assumption is manifest in Theorems 5.11 and 5.12 since the hypotheses Π in the two judgments $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow s \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$ and $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u, u]$ in the statements of the theorems are identical. Practically, this translates to the assumption that a policy rule once issued can never be revoked or canceled, except perhaps due to expiration at the time mentioned in the rule which is handled automatically by the BL inference system. In reality, a policy rule may be issued in error, and therefore, a mechanism for untimely revocation is essential. In this section we discuss a straightforward mechanism for enforcing revocation in the PCFS architecture. Although not supported in the current implementation of PCFS, it can be easily added.

Let us assume that revoked policy certificates are explicitly available to the PCFS back end as a list. This is not an unreasonable assumption since most existing certificate schemes allow for such lists, often called Certificate Revocation Lists (CRLs). (The issue of who can create the entries in such a list is orthogonal to our description of enforcement of revocation; usually the creator of a certificate is allowed to revoke it.) Both formally and in practice, a CRL can be represented as a list $\mathcal{R} = \pi_1, \ldots, \pi_n$ of proof variables that name revoked policy rules. Assuming that the CRL available to the PCFS back end is always current, revocation may be enforced in PCFS using the following two-step procedure.

R1. At the time of proof verification, the PCFS verification tool writes the list $\mathcal{L}$ of all proof variables that appear free in the proof term V it checks into the procap it generates ($\mathcal{L}$ becomes an additional condition like $\mathcal{C}$ and $\mathcal{I}$).

R2. Before accepting any procap, the back end of PCFS checks that $\mathcal{L} \cap \mathcal{R} = \phi$, where $\mathcal{L}$ is the list of proof variables included in the procap and $\mathcal{R}$ is the CRL available to the back end.

To show that this two-step procedure implements revocation soundly, it suffices to establish that whenever a procap is accepted by the back end of PCFS, it is the case that the proof term from whose checking the procap was derived can also be checked in a hypotheses that is *valid at the time the procap is accepted.* To establish this result, we need a few definitions and a lemma.

Definition 5.13. Define $\Pi \backslash \mathcal{R}$ to be the hypotheses $\{(\pi : J) \in \Pi \mid \pi \notin \mathcal{R}\}$.

Definition 5.14. $\mathtt{fpv}(V)$ and $\mathtt{fpv}(R)$ denote the sets of proof variables occurring free in V and R respectively.

Lemma 5.15 (Strengthening)**.** *The following hold.*

A. If $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *and* $\mathtt{fpv}(V) \cap \mathcal{R} = \phi$*, then* $\Sigma; \Psi; E; \Pi \backslash \mathcal{R} \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$.

B. If $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *and* $\mathtt{fpv}(V) \cap \mathcal{R} = \phi$*, then* $\Sigma; \Psi; E; \Pi \backslash \mathcal{R} \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$.

Proof. By simultaneous induction on derivations given in A and B and case analysis of their last rules. □

Theorem 5.16 (Soundness of revocation enforcement)**.** *Suppose that a procap obtained by verifying* $\Sigma;\cdot;\cdot;\Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C};\mathcal{I}$ *is accepted by the PCFS back end. Then there is a hypotheses* Π' *valid when the procap is accepted such that* $\Sigma;\cdot;\cdot;\Pi' \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C};\mathcal{I}$.

Proof. Let $\mathcal{R}$ be the revocation list when the procap is accepted, and let $\mathcal{L} = \mathtt{fpv}(V)$. Since step R2 must execute successfully in order for the procap to be accepted, it follows that $\mathcal{L} \cap \mathcal{R} = \phi$, or equivalently, $\mathtt{fpv}(V) \cap \mathcal{R} = \phi$. Hence by Lemma 5.15(A), $\Sigma;\cdot;\cdot;\Pi\backslash\mathcal{R} \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C};\mathcal{I}$. Choose $\Pi' = \Pi\backslash\mathcal{R}$. Due to our assumption that the CRL available to the back end is current, every hypothesis in Π' must also be valid when the procap is accepted. □

To show that the enforcement of revocation is complete, we must argue that the procap generated from verification of a proof term will be accepted by the PCFS back end if no proof variables occurring in the proof term have been revoked (and the other conditions $\mathcal{C}$ and $\mathcal{I}$ of the procap hold). This is trivially true – if no proof variables occurring in the proof term have been revoked, then $\mathcal{L} \cap \mathcal{R}$ in step R2 is ϕ by definition of $\mathcal{L}$ and $\mathcal{R}$.

5.3 Proof Terms from the Sequent Calculus

Even though we have discussed proof terms for natural deduction so far, it is reasonable to expect that any automated proof search tool for BL will be based on the sequent calculus of §4.2.4 (this includes the proof search tool provided with PCFS; see §6). The objective of this section is to explain how proof terms may be derived from a sequent calculus proof. Given that there are constructive proofs which show both that the sequent calculus can be simulated in natural deduction (Theorem 4.14) and that proof terms can be assigned to natural deduction proofs (Theorem 5.3), a procedure to derive proof terms from sequent calculus proofs may be obtained simply by composing the proofs of the two theorems. However, keeping in mind that such a procedure is central to the implementation of proof search tools, it is useful to understand the composed procedure in the simplest terms. This is precisely the objective of this section – it describes a direct assignment of proof terms to sequent calculus proofs, without a detour into natural deduction.

The direct assignment of proof terms to sequent calculus proofs is described as a new inference system, which only adds proof annotations to the sequent calculus of Figures 4.4 and 4.5. This annotated sequent calculus is shown in Figures 5.5 and 5.6. Its hypothetical judgment, called an annotated sequent, has the form $\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$. The rules of the annotated sequent calculus correspond one-to-one with the rules of the sequent calculus and preserve the structure of the latter. Proof terms are constructed using the following general rules.

- For a sequent calculus proof ending in a right rule, a proof term is obtained using the corresponding introduction constructor of checkables (e.g., rule (saysR)).

$$\frac{\Sigma;\Psi \models u_1' \leq u_1 \qquad \Sigma;\Psi \models u_2 \leq u_2'}{\Sigma;\Psi;E;\Pi,\pi : p \circ [u_1',u_2'] \xrightarrow{\nu} \pi : p \circ [u_1,u_2]}\text{init}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi,\pi : k\ \mathsf{claims}\ s \circ [u_1,u_2],\tau : s \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2'] \\ \nu = k',u_b,u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2 \qquad \Sigma;\Psi \models k \succeq k'\end{array}}{\Sigma;\Psi;E;\Pi,\pi : k\ \mathsf{claims}\ s \circ [u_1,u_2] \xrightarrow{\nu} V[\pi/\tau] : r \circ [u_1',u_2']}\text{claims}$$

$$\frac{\Sigma;\Psi;E;\Pi| \xrightarrow{k,u_1,u_2} V : s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \mathtt{saysI}\ V : k\ \mathsf{says}\ s \circ [u_1,u_2]}\mathsf{saysR}$$

$$\frac{\Sigma;\Psi;E;\Pi,\pi : k\ \mathsf{says}\ s \circ [u_1,u_2],\tau : k\ \mathsf{claims}\ s \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi,\pi : k\ \mathsf{says}\ s \circ [u_1,u_2] \xrightarrow{\nu} \mathtt{saysE}\ \pi\ (\tau.V) : r \circ [u_1',u_2']}\mathsf{saysL}$$

$$\frac{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \mathtt{atI}\ V : s\ @\ [u_1,u_2] \circ [u_1',u_2']}@\text{R}$$

$$\frac{\Sigma;\Psi;E;\Pi,\pi : s\ @\ [u_1',u_2'] \circ [u_1,u_2],\tau : s \circ [u_1',u_2'] \xrightarrow{\nu} V : r \circ [u_1'',u_2'']}{\Sigma;\Psi;E;\Pi,\pi : s\ @\ [u_1',u_2'] \circ [u_1,u_2] \xrightarrow{\nu} \mathtt{atE}\ \pi\ (\tau.V) : r \circ [u_1'',u_2'']}@\text{L}$$

$$\frac{\Sigma;\Psi \models c}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \mathtt{consI} : c \circ [u_1,u_2]}\text{consR}$$

$$\frac{\Sigma;\Psi,c;E;\Pi,\pi : c \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi,\pi : c \circ [u_1,u_2] \xrightarrow{\nu} \mathtt{consE}\ \pi\ V : r \circ [u_1',u_2']}\text{consL}$$

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \mathtt{interI} : i \circ [u_1,u_2]}\text{interR}$$

$$\frac{\Sigma;\Psi;E,i;\Pi,\pi : i \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi,\pi : i \circ [u_1,u_2] \xrightarrow{\nu} \mathtt{interE}\ \pi\ V : r \circ [u_1',u_2']}\text{interL}$$

Figure 5.5: Annotated sequent calculus, part 1

- If a sequent calculus proof ends in a left rule then the proof term is obtained by naming the component(s) of the principal judgment by new variable(s) in the premises and either substituting these variables in the conclusion (e.g., the rule ($\wedge$L)) or binding the new variables (e.g., the rule (saysL)), in each case with an elimination constructor for proof terms.

The rules of Figures 5.5 and 5.6 do not mention the constructor `check`. Since `check` is the only constructor that takes a formula as an argument, it follows that any proof term constructed from a sequent calculus does not contain any formula as an annotation. We

$$\frac{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V_1 : s_1 \circ [u_1,u_2] \qquad \Sigma;\Psi;E;\Pi \xrightarrow{\nu} V_2 : s_2 \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{conjI}\ V_1\ V_2 : s_1 \wedge s_2 \circ [u_1,u_2]}\wedge\mathrm{R}$$

$$\frac{\Sigma;\Psi;E;\Pi,\pi : s_1 \wedge s_2 \circ [u_1,u_2],\tau_1 : s_1 \circ [u_1,u_2],\tau_2 : s_2 \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi,\pi : s_1 \wedge s_2 \circ [u_1,u_2] \xrightarrow{\nu} V[(\texttt{conjE1}\ \pi)/\tau_1][(\texttt{conjE2}\ \pi)/\tau_2] : r \circ [u_1',u_2']}\wedge\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s_1 \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{disjI1}\ V : s_1 \vee s_2 \circ [u_1,u_2]}\vee\mathrm{R}_1 \qquad \frac{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s_2 \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{disjI2}\ V : s_1 \vee s_2 \circ [u_1,u_2]}\vee\mathrm{R}_2$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi,\pi : s_1 \vee s_2 \circ [u_1,u_2],\tau_1 : s_1 \circ [u_1,u_2] \xrightarrow{\nu} V_1 : r \circ [u_1',u_2'] \\ \Sigma;\Psi;E;\Pi,\pi : s_1 \vee s_2 \circ [u_1,u_2],\tau_2 : s_2 \circ [u_1,u_2] \xrightarrow{\nu} V_2 : r \circ [u_1',u_2']\end{array}}{\Sigma;\Psi;E;\Pi,\pi : s_1 \vee s_2 \circ [u_1,u_2] \xrightarrow{\nu} \texttt{disjE}\ \pi\ (\tau_1.V_1)\ (\tau_2.V_2) : r \circ [u_1',u_2']}\vee\mathrm{L}$$

$$\frac{}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{topI} : \top \circ [u_1,u_2]}\top\mathrm{R} \qquad \frac{}{\Sigma;\Psi;E;\Pi,\pi : \bot \circ [u_1,u_2] \xrightarrow{\nu} \texttt{botE}\ \pi : r \circ [u_1',u_2']}\bot\mathrm{L}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time};\Psi,u_1 \le x_1, x_2 \le u_2;E;\Pi,\pi : s_1 \circ [x_1,x_2] \xrightarrow{\nu} V : s_2 \circ [x_1,x_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) : s_1 \supset s_2 \circ [u_1,u_2]}\supset\mathrm{R}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Pi,\pi : s_1 \supset s_2 \circ [u_1,u_2] \xrightarrow{\nu} V_1 : s_1 \circ [u_1',u_2'] \\ \Sigma;\Psi;E;\Pi,\pi : s_1 \supset s_2 \circ [u_1,u_2],\tau : s_2 \circ [u_1',u_2'] \xrightarrow{\nu} V_2 : r \circ [u_1'',u_2''] \\ \Sigma;\Psi \models u_1 \le u_1' \qquad \Sigma;\Psi \models u_2' \le u_2\end{array}}{\Sigma;\Psi;E;\Pi,\pi : s_1 \supset s_2 \circ [u_1,u_2] \xrightarrow{\nu} V_2[(\texttt{impE}\ \pi\ V_1\ u_1'\ u_2')/\tau] : r \circ [u_1'',u_2'']}\supset\mathrm{L}$$

$$\frac{\Sigma, x{:}\sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s \circ [u_1,u_2]}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{forallI}\ (x.V) : \forall x{:}\sigma.s \circ [u_1,u_2]}\forall\mathrm{R}$$

$$\frac{\Sigma;\Psi;E;\Pi,\pi : \forall x{:}\sigma.s \circ [u_1,u_2],\tau : s[t/x] \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2'] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Psi;E;\Pi,\pi : \forall x{:}\sigma.s \circ [u_1,u_2] \xrightarrow{\nu} V[(\texttt{forallE}\ t\ \pi)/\tau] : r \circ [u_1',u_2']}\forall\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} V : s[t/x] \circ [u_1,u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Psi;E;\Pi \xrightarrow{\nu} \texttt{existsI}\ t\ V : \exists x{:}\sigma.s \circ [u_1,u_2]}\exists\mathrm{R}$$

$$\frac{\Sigma, x{:}\sigma;\Psi;E;\Pi,\pi : \exists x{:}\sigma.s \circ [u_1,u_2],\tau : s \circ [u_1,u_2] \xrightarrow{\nu} V : r \circ [u_1',u_2']}{\Sigma;\Psi;E;\Pi,\pi : \exists x{:}\sigma.s \circ [u_1,u_2] \xrightarrow{\nu} \texttt{existsE}\ (x.\tau.V) : r \circ [u_1',u_2']}\exists\mathrm{L}$$

Figure 5.6: Annotated sequent calculus, part 2

make a similar observation for proof terms derived from canonical proofs in §5.4.

If all proof variables and proof terms are erased from a derivation in the annotated calculus, then a correct sequent calculus derivation is obtained. Dually, given a sequent calculus derivation, it is easy to annotate it with proof terms following the rules of Figures 5.5 and 5.6. These observations are formalized in the following two theorems.

Theorem 5.17 (Soundness)**.** *If* $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$*, then* $\Sigma; \Psi; E; |\Pi| \xrightarrow{\nu} s \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$. The idea is to systematically erase all proof variables and proof terms from the given derivation. □

Theorem 5.18 (Completeness)**.** *If* $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *and* $|\Pi| = \Gamma$*, then there is a checkable* V *such that* $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$. □

Further, every proof term constructed by the annotated sequent calculus is "correct" in the sense that it can be checked using the rules of Figures 5.1 and 5.2.

Theorem 5.19 (Correctness)**.** *If* $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$*, then* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$ and case analysis of its last rule. See Appendix C, Theorem C.6 for some representative cases. □

Theorems 5.18 and 5.19 together imply that the rules of Figures 5.5 and 5.6 constitute a sound and complete proof term assignment system for the sequent calculus of BL. This proof term assignment system is used in the proof search tool described in §6, although for simplicity, that chapter does not mention proof terms. Also, Theorems 5.18, 5.19, and 5.2 together provide another proof that the sequent calculus of BL can be simulated in natural deduction (Theorem 4.14).

5.4 Proof Terms for Canonical Proofs

The subject of this section is orthogonal to the rest of the thesis and the disinclined reader may skip it without a break in continuity.

As observed in §5.1, the rules of Figures 5.1 and 5.2 are very similar to those that define canonical and atomic proofs for BL (Figures 4.7 and 4.8). In fact the proof term calculus excluding both the proof term constructor `check` and the rule (check) is a proof term assignment for canonical and atomic proofs. Precisely, if a derivation in the proof term system does not contain the rule (check), then erasing all proof variables and proof terms, replacing each $\Longleftarrow \cdot$ by $\cdot \Uparrow$, and $\Longrightarrow \cdot$ by $\cdot \Downarrow$ results in a valid derivation by the rules of Figures 4.7 and 4.8. Dually, each canonical proof is witnessed by a checkable, and each atomic proof by an inferable. These observations are formalized in the following theorems.

Theorem 5.20 (Soundness)**.** *The following hold for V and R that do not contain the constructor* `check`*.*

1. *If* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ *then* $\Sigma; \Psi; E; |\Pi| \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$*.*
2. *If* $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ *then* $\Sigma; \Psi; E; |\Pi| \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$*.*

Proof. By simultaneous induction on the given derivations. □

Theorem 5.21 (Completeness)**.** *Suppose* $|\Pi| = \Gamma$*. Then the following hold.*

1. *If* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$*, then there is a checkable V such that* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$*.*
2. *If* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$*, then there is an inferable R such that* $\Sigma; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$*.*

Further the V in (1) and the R in (2) can be chosen such that they do not contain the constructor `check`*.*

Proof. By simultaneous induction on the derivations given in (1) and (2). □

Proof term normalization. Given that proof terms without the constructor `check` correspond to canonical and atomic proofs, we may define *canonical and atomic proof terms* as follows.

- A checkable V is called canonical if it does not contain the constructor `check` in it.
- An inferable R is called atomic if it does not contain the constructor `check` in it.

Canonical proof terms are BL analogues of the usual β-normal forms in the typed lambda calculus since they cannot contain a β-redex. (From the syntax of proof terms, the principal argument of every elimination constructor in a canonical proof term must have at the top level an elimination constructor).

Further, Theorems 5.2, 4.19, and 5.21 imply that *any* proof term which witnesses a judgment can be "normalized". Suppose that V is a checkable such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$. By Theorem 5.2, $\Sigma; \Psi; E; |\Pi| \vdash^{\nu} s \circ [u_1, u_2]$ and by Theorem 4.19, $\Sigma; \Psi; E; |\Pi| \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$. Therefore by Theorem 5.21 there is a canonical proof term V' such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V' \Longleftarrow s \circ [u_1, u_2]$. What is more interesting here is that the proofs of Theorems 5.2, 4.19, and 5.21 can be composed to obtain a normalization procedure that produces a canonical V' from V. However, owing to our use of admissibility of cut in the proof of Theorem 4.19, this procedure is somewhat indirect and provides little insight beyond what is already present in the proofs of Theorems 5.2, 4.19, and 5.21. Since we have no occasion to use the normalization procedure in this thesis, we do not present it in any further detail. The same normalization result can also be established through reduction rules for proof terms.

5.5 Related Work

There is a significant amount of related work on proof terms for intuitionistic logics, as well as a limited amount of related work on proof terms in the context of authorization. We start with a description of the former and then turn to the latter.

Broadly speaking, our presentation of proof terms derives from the well known Curry-Howard isomorphism which states that the typed lambda calculus is a proof term assignment system for intuitionistic logic (see, e.g., [72]). Extensions of the isomorphism are known for variants of intuitionistic logics including linear logic [141] and modal logics [115], as well as for classical logic [75]. More specifically, the bidirectional style of proof terms as well as the verification procedures of Theorem 5.8 are based on prior work on bidirectional type systems [117], and are even more closely related to Pfenning's notes on proof systems for intuitionistic logic [114]. In particular, the assignment of proof terms to sequent calculus proofs and canonical proofs of BL is a generalization of similar work in the latter.

In the context of authorization, DeYoung investigates bidirectional proof terms for η logic in his undergraduate thesis [53]. The syntax of proof terms presented in §5.1 is similar to and based on that work. In particular, the specific interactions between inference, checking, and explicit time in the rules (infer) and (check) of Figure 5.1 go back to DeYoung's work. DeYoung also describes an implementation of proof verification procedures analogous to those in Theorem 5.8. The two-part verification procedure that PCFS implements, its formalization, and its correctness are all novel to this thesis. Since DeYoung's work was not designed for a realistic implementation, it does not discuss properties of proof terms although it seems that analogues of all properties from §5.1.2 should also hold for proof terms of η logic.

On a more practical note, the implementation of the authorization policy language SecPAL includes a proof visualization tool which can be used to inspect graphically the structure of an inference [23]. The tool may very useful for administrators and users, both for debugging policies as well as for audit.

Prior work by Chaudhuri also considers mechanisms for enforcement of authorization policies by offlining policy decisions and using capabilities to convey the results of decisions to the reference monitor [40]. However, both policy decisions and capabilities are kept abstract. While some of the ideas in that work are similar to those in §5.2.2, e.g., Chaudhuri's correctness theorems also compare capability based enforcement mechanisms to ideal ones where the policy decision is made by the reference monitor, the problems addressed in that work and ours are different. We work with a fixed logic and show that the process of extracting conditions from proofs and checking them in the back end is sound and complete. On the other hand, Chaudhuri's work is concerned with ensuring that the architecture with capabilities is observationally similar to that without them. As a result, certain issues like information leaks on denied access are relevant in Chaudhuri's work, not ours. Similarly, the effect of state changes on policies is relevant only in PCFS, not Chaudhuri's work. Aside from such differences, Chaudhuri's work has had a significant influence on the high-level design of PCFS.

Chapter 6

BL: Goal-directed Proof Search

In this chapter we discuss the theory and implementation of a practical method for automatically finding proofs of authorization given policy rules represented in our logic BL. This method is the basis of the proof search tool `pcfs-search` that is included with our file system PCFS (§7.1). Broadly construed, the proof search method is based on the sequent calculus (§4.2.4). However, the sequent calculus by itself is too non-deterministic to be used for proof search directly – a proof of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ may be obtained by applying either a right rule to s, or a left rule to any of the hypotheses in Γ. This results in a large number of choices at each step, due to which any proof search based naively on the sequent calculus may have to explore a formidably large space of proofs. The approach we follow in this chapter, called goal-directed search, reduces this search space significantly (goal-directed search is explained in §6.1). To maintain completeness with respect to the sequent calculus, we restrict our attention to an expressive fragment of BL called BL_G. All policies presented in this thesis lie in this fragment.

Precisely, the problem we are trying to address in this chapter is that of finding a proof term V such that $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$, if such a proof term exists. We assume that Σ, Ψ, E, Π, ν, s, u_1, and u_2 are given. As discussed in §7.1, these inputs are obtained from various sources in the PCFS proof search tool. In particular, the user invoking the tool is responsible for providing s, u_1, u_2, and Π (the latter is in the form of digital certificates). However, for simplicity, we omit a description of proof terms from this chapter, and concentrate only on constructing the proof using inference rules. Proof terms may be added easily to the proof search calculus described here using ideas from assignment of proof terms to the sequent calculus (§5.3), which is also what `pcfs-search` does.

The rest of this chapter is organized as follows. In §6.1 we present background material on goal-directed proof search. §6.2 presents the fragment BL_G, and the rules for goal-directed proof search. In §6.3, we prove that goal-directed proof search is sound and complete with respect to the sequent calculus. §6.4 discusses implementation-specific issues that are relevant to the tool `pcfs-search`. Related work is presented in §6.5. It is important to clarify that the method of proof search presented in this chapter is not a decision procedure, since BL_G is not decidable. Completeness of proof search means that there is a non-deterministic strategy within the confines of the formal rules of goal-directed proof search

that will find a proof if one exists. The strategy may be determinized using breadth-first search, thus making proof search a semi-decision procedure, although this is often slow in practice, so `pcfs-search` uses depth-first search with backtracking instead.

We advise readers to revisit the sequent calculus for BL presented in §4.2.4 before reading the rest of this chapter.

6.1 Background: What Is Goal-directed Proof Search?

The term "goal" refers to the formula in the conclusion of a sequent that we wish to establish through proof search. Precisely, the goal of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ is s. By goal-directed search we mean a backwards[1] search for a proof which proceeds at each step by decomposing the goal of the sequent, in accordance with the right rules of the sequent calculus. Other rules are used only when the goal is atomic (in the specific case of BL, only when the conclusion has the form $p \circ [u_1, u_2]$). For example, to find a proof of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_1, u_2]$, goal-directed search would try to find proofs of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]$, and then combine them with the rule ($\wedge$R). Similarly, to find a proof of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \vee s_2 \circ [u_1, u_2]$ goal-directed search would try to find a proof of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2]$, and if it fails, it would try to find a proof of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]$. (The attempts could also be made in the other order.) If either case succeeds the proof would be completed using the corresponding rule ($\vee\mathrm{L}_1$) or ($\vee\mathrm{L}_2$), else the proof search would fail.

If the goal is atomic, or more precisely the conclusion has the form $p \circ [u_1, u_2]$ in BL, then goal-directed search proceeds by *backchaining*. The idea of backchaining, based on languages like Prolog, is to pick a suitable hypothesis $s' \circ [u_1', u_2'] \in \Gamma$ and to use it to find a list of judgments $\vec{J}$ such that provability of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} J$ for each $J \in \vec{J}$ implies provability of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} p \circ [u_1, u_2]$. The elements of $\vec{J}$ are often called subgoals. The proof search procedure then tries to find proofs of subgoals recursively, which, if successful, entail provability of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} p \circ [u_1, u_2]$. Formal rules for finding $\vec{J}$ given $s' \circ [u_1', u_2']$ and $p \circ [u_1, u_2]$ are loosely based on the left rules of the sequent calculus, but the connection between the two is not as strong as the connection between the right rules of the sequent calculus and the rules for decomposing goals in goal-directed search. Backchaining for BL is covered in §6.2.

Whereas goal-directed search constrains non-determinism in the sequent calculus significantly, and is both easy to formalize and simple to implement efficiently, there is an important theoretical concern that must be addressed: Is goal-directed search *complete* in the sense that if a sequent is provable then goal-directed search will find a proof of it? It is quite easy to show that this is not true because for certain kinds of goals, it is not the case that if a sequent with the goal is provable, then the sequents obtained by applying one of the right rules *backwards* would also be provable. For example, provability of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu}$

[1] The adjective "backwards" means that the rules of the sequent calculus are applied from the conclusion to the premises, thus constructing a proof from the sequent to be proved towards the leaves. The other possibility, which we do not consider in this chapter, is to apply the rules from premises to conclusions. The latter is called the inverse method.

$s_1 \vee s_2 \circ [u_1, u_2]$ does not, in general, imply that one of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2]$ and $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]$ is provable. Consequently, goal-directed search may not always succeed on a provable sequent whose goal has a top level disjunction in it. Similar incompleteness exists for the connectives $\bot$, $\exists$, and says.

Given that goal-directed search is incomplete for all of BL (and similarly for all of intuitionistic logic), the next question is whether there is a useful fragment of the logic for which goal-directed search is complete. The answer to this question is affirmative. Most logics (including BL) have large fragments on which goal-directed search is complete. The common idea in these fragments is to restrict the occurrences of connectives in goals and hypotheses. Interestingly, and perhaps non-intuitively, such restrictions invariably imply that whenever a sequent is provable, then the sequents obtained by applying backwards one of the right rules of the top level connective of its goal are all provable, thus making goal-directed search complete. As illustrations, we list below, in increasing order of expressiveness, some previously investigated fragments of intuitionistic and linear logic on which goal-directed search is complete.

- The well-known Horn fragment of first-order logic, which restricts connectives in hypotheses to atoms, $\wedge$, $\top$, $\supset$, and $\forall$, and those in goals to atoms, $\wedge$, $\top$, $\vee$, $\bot$, and $\exists$ (see, e.g., [103]). Prolog is based on this fragment.

- The Hereditary-Harrop fragment, which generalizes the Horn fragment by allowing all connectives in goals [103]. The main difference between the Horn fragment and the Hereditary-Harrop fragment is that in the former the hypotheses do not change during proof search whereas in the latter they may change due to backwards applications of the rule ($\supset$R).

- The linear logic programming language Lolli [78], which in addition to including linear connectives, generalizes the Hereditary-Harrop fragment by allowing the linear analogues of the connectives $\vee$, $\bot$, and $\exists$ at the top level in hypotheses. This fragment requires a proof search procedure that is slightly more general than goal-directed search proper, but still fits into the general paradigm of "goal-directed". Precisely, whenever a new hypothesis is introduced using the rule ($\supset$R), its top level $\vee$ and $\exists$ connectives are immediately decomposed with left rules, after which goal decomposition is resumed.

Salient points about this chapter. The objective of this chapter is to present a fragment of BL for which goal-directed search is complete, explain formally via inference rules goal-directed proof search for it, prove that the goal-directed search is both sound and complete with respect to the rules of the sequent calculus restricted to the fragment, and explain its implementation in the proof search tool `pcfs-search` included in PCFS. We call this fragment BL_G. It is based on ideas from the linear logic programming languages Lolli [78] and LolliMon [98]. Like Lolli, the proof search procedure for BL_G is a slight generalization of goal-directed search proper (this generalization was described in the third point above). We make three salient observations before presenting the technical material.

First, despite its syntactic restrictions, BL_G is very expressive. In particular, all policies presented in this thesis, including all policies of the case study (§8), lie in BL_G.

Second, goal-directed search is an instance of a general proof search technique called focusing [12]. Although this technique is compatible with BL, and may be used to construct a generic theorem prover for all of BL, we refrain from doing so here because such a generic tool would be needlessly complicated and possibly slow because it would have to cater to many cases that may never arise in actual policies.

Third, both the connective says and explicit time, and the latter in particular, increase significantly the complexity of goal-directed search as well as the proof of its soundness and completeness. As a result, prior work on goal-directed search for other logics does not apply directly, although the methods described in earlier work are certainly the basis of the technical development of this chapter [78, 103].

6.2 Goal-directed Proof Search in BL_G

The syntax of the fragment BL_G is described below. Goals g are formulas that may appear in conclusions of sequents in the fragment. They are allowed to contain all connectives. Formulas in hypotheses are divided into two categories: (1) *Clauses* d that contain only uninterpreted atoms and the connectives $\supset$, $\wedge$, $\top$, $\forall$, and @, and (2) *Chunks* h that may contain clauses d, constraints c, interpreted connectives i, and formulas of the form k says d at their leaves and the connectives $\wedge$, $\vee$, $\top$, $\bot$, $\exists$, and @. Hypotheses also take two forms: *policies* Δ that contain assumptions of the form $d \circ [u_1, u_2]$ and k claims $d \circ [u_1, u_2]$, and *groups* Ξ that have assumptions of the form $h \circ [u_1, u_2]$. Unlike all other hypotheses considered in this thesis so far, a group is an ordered list, not a multiset. We represent groups using familiar notation for lists – [] denotes an empty group, and $\Xi :: (h \circ [u_1, u_2])$ denotes the group obtained by adding $h \circ [u_1, u_2]$ to the end of Ξ.

$$
\begin{array}{llll}
\text{Goals} & g & ::= & p \mid c \mid i \mid g_1 \wedge g_2 \mid g_1 \vee g_2 \mid h \supset g \mid \top \mid \bot \mid \forall x{:}\sigma.g \mid \exists x{:}\sigma.g \mid \\
 & & & k \text{ says } g \mid g \mathbin{@} [u_1, u_2] \\
\text{Clauses} & d & ::= & p \mid d_1 \wedge d_2 \mid \top \mid g \supset d \mid \forall x{:}\sigma.d \mid d \mathbin{@} [u_1, u_2] \\
\text{Chunks} & h & ::= & d \mid c \mid i \mid h_1 \wedge h_2 \mid h_1 \vee h_2 \mid \top \mid \bot \mid \exists x{:}\sigma.h \mid k \text{ says } d \mid h \mathbin{@} [u_1, u_2] \\
 & & & \\
\text{Policies} & \Delta & ::= & \cdot \mid \Delta, d \circ [u_1, u_2] \mid \Delta, k \text{ claims } d \circ [u_1, u_2] \\
\text{Groups} & \Xi & ::= & [] \mid \Xi :: (h \circ [u_1, u_2])
\end{array}
$$

Our objective now is to define a calculus that formalizes goal-directed search for finding proofs of sequents of the form $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g \circ [u_1, u_2]$. We describe this calculus using five kinds of hypothetical judgments – labeled R, Q, L, N, F – whose forms are listed below. At the expense of overloading terminology, we use the term "sequent" to refer to these hypothetical judgments as well. Whether the term sequent refers to sequents of the sequent calculus, or hypothetical judgments of goal-directed search should always be clear from the

context.

$$
\begin{array}{lllll}
\text{Query lists} & \mathcal{Q} & ::= & [] \mid (u_1 \leq u_2) :: \mathcal{Q} \mid (g \circ [u_1, u_2]) :: \mathcal{Q} \\
\text{Boolean flags} & b & ::= & \mathtt{tt} \mid \mathtt{ff} \\
\text{Sequents} & \mathrm{R} & ::= & \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2] \\
 & \mathrm{Q} & ::= & \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q} \\
 & \mathrm{L} & ::= & \Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2] \\
 & \mathrm{N} & ::= & \Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2] \\
 & \mathrm{F} & ::= & \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b
\end{array}
$$

Although the exact rules of goal-directed search that may be used to establish these five forms of sequents are described in §6.2.1, we briefly explain the purpose and meaning of each form of sequent here. In R-sequents $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$, the goal $g \circ [u_1, u_2]$ is decomposed by rules similar to the right rules of the sequent calculus. These are the primary form of sequents in which proof search starts and they also justify the adjective "goal-directed". Q-sequents $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ are a generalization of R-sequents in which the goal is replaced by a query list $\mathcal{Q}$, which may contain conclusions of the forms $u_1 \leq u_2$ and $g \circ [u_1, u_2]$. A query list is the BL analogue of subgoals mentioned in §6.1 in the context of backchaining. The meaning of $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ is that a proof of every conclusion in $\mathcal{Q}$ can be found from the hypotheses $\Sigma; \Psi; E; \Delta$.

In L-sequents $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$, the chunks in Ξ are decomposed using rules similar to the left rules of the sequent calculus. These sequents arise from R-sequents when a hypothesis is introduced by decomposition of an implication in a goal. N-sequents $\Sigma; \Psi; E\Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ are the site of backchaining. To prove an N-sequent, we choose a clause from the hypotheses that can be used to prove the atomic goal $p \circ [u_1, u_2]$, decompose the clause in an F-sequent, and prove the resulting query list (subgoals). An F-sequent $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ is used to decompose the clause $d \circ [u_1, u_2]$ with the aim of proving $p \circ [u_1', u_2']$; it means that the latter is entailed by the former if in addition every conclusion in the query list $\mathcal{Q}$ holds. The boolean b in an F-sequent is a flag that is needed to correctly account for time intervals, as explained in §6.2.1.

Query lists in the sequent calculus. In order to prove soundness and completeness of goal-directed search (§6.3), it is helpful to define an auxiliary judgment $\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} \mathcal{Q}$ based on the rules of the sequent calculus (not goal-directed search) as shown below. This judgment is analogous to the judgment $\Sigma; \Psi; E; \Gamma \overset{\nu}{\Rightarrow} \mathcal{Q}$ for goal-directed search.

$$\frac{}{\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} []}[]\mathrm{Q} \qquad \frac{\Sigma; \Psi \models u_1 \leq u_2 \qquad \Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} (u_1 \leq u_2) :: \mathcal{Q}}\mathrm{leqQ}$$

$$\frac{\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} g \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} (g \circ [u_1, u_2]) :: \mathcal{Q}}\mathrm{goalQ}$$

	Sequent of goal-directed search	Sequent calculus analogue
R	$\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$	$\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrowtail} g \circ [u_1, u_2]$
Q	$\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$	$\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrowtail} \mathcal{Q}$
L	$\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$	$\Sigma; \Psi; E; \Delta, \lvert\Xi\rvert \overset{\nu}{\rightarrowtail} g \circ [u_1, u_2]$
N	$\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$	$\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrowtail} p \circ [u_1, u_2]$
F	$\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$	No direct analogue. Means that $\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrowtail} \mathcal{Q}$ implies $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \overset{\nu}{\rightarrowtail} p \circ [u_1', u_2']$

Figure 6.1: Correspondence between sequents of goal-directed search and the sequent calculus

The above rules induct over the query list $\mathcal{Q}$ in $\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrowtail} \mathcal{Q}$ and check that each conclusion in $\mathcal{Q}$ is provable from the hypotheses $\Sigma; \Psi; E; \Gamma$.

To help the reader get a better idea of the meanings of the sequents used in goal-directed search, we list in Figure 6.1 the form of sequents from BL's sequent calculus that correspond to the sequent classes R, Q, L, and N. F-sequents have no direct analogues in the sequent calculus, but their meaning can be explained in terms of entailments between sequents of the latter. The notation $|\Xi|$ denotes the multiset obtained by ignoring the order of elements in Ξ.

6.2.1 Rules of Goal-directed Proof Search

The rules for goal-directed search are shown in Figures 6.2, 6.3, and 6.4. All rules are used backwards during proof search.

R-sequents. Search starts in an R-sequent $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$. The rules applying to an R-sequent (Figure 6.2) are similar to the right rules of the sequent calculus, and always decompose the goal. Unlike the sequent calculus, where left and right rules may be arbitrarily interleaved, R-sequents force that the goal be continuously decomposed till a leaf – uninterpreted atom, interpreted atom, or constraint – is reached. The only exception to this forced decomposition of the goal is the rule (R-$\supset$), which introduces a chunk in the hypothesis in the premise, and therefore transitions temporarily to an L-sequent where the chunk is decomposed by left rules (described later). After the chunk is decomposed completely, the procedure returns to an R-sequent, and decomposition of the goal continues. If a constraint is reached, the constraint solver is invoked to attempt to solve it (rule (R-cons)). Interpreted predicates are similarly treated by the rule (R-inter). If an uninterpreted atom is reached, the proof search transitions to an N-sequent (rule (R-N)), where backchaining is used to try to prove the atomic goal (described later).

The reason why such forced decomposition of goals is complete with respect to the sequent calculus is that once hypotheses are restricted to the form Δ, the right rules of all connectives in the sequent calculus become invertible in the following sense: whenever $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrowtail} g \circ [u_1, u_2]$ is provable in the sequent calculus, then the premises of at

R-sequents

$$\frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} p \circ [u_1, u_2]}\text{R-N} \qquad \frac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} c \circ [u_1, u_2]}\text{R-cons}$$

$$\frac{\Sigma; E \models i}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} i \circ [u_1, u_2]}\text{R-inter} \qquad \frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \wedge g_2 \circ [u_1, u_2]}\text{R-}\wedge$$

$$\frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]}\text{R-}\vee_1 \qquad \frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]}\text{R-}\vee_2$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Delta; h \circ [x_1, x_2] \overset{\nu}{\Leftarrow} g \circ [x_1, x_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} h \supset g \circ [u_1, u_2]}\text{R-}\supset$$

$$\frac{}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \top \circ [u_1, u_2]}\text{R-}\top \qquad \frac{\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \forall x{:}\sigma.g \circ [u_1, u_2]}\text{R-}\forall$$

$$\frac{\Sigma \vdash t : \sigma \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g[t/x] \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \exists x{:}\sigma.g \circ [u_1, u_2]}\text{R-}\exists \qquad \frac{\Sigma; \Psi; E; \Delta |\overset{k, u_1, u_2}{\Rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} k \ \mathsf{says}\ g \circ [u_1, u_2]}\text{R-}\mathsf{says}$$

$$\frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]}\text{R-@}$$

Q-sequents

$$\frac{}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} []}\text{Q-[]} \qquad \frac{\Sigma; \Psi \models u_1 \leq u_2 \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (u_1 \leq u_2) :: \mathcal{Q}}\text{Q-leq}$$

$$\frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (g \circ [u_1, u_2]) :: \mathcal{Q}}\text{Q-goal}$$

Figure 6.2: Goal-directed search, part 1

least one of the right rules that could be used directly to prove this sequent can also be established. Even though we do not use this fact in the proof of completeness of proof search directly (Theorem 6.5), we have included a formal statement of the fact in Appendix D, Lemmas D.16 and D.17 for the curious readers.

Note that proof search may have to explore multiple branches, as would happen after application of the rule (R-$\wedge$) which has two premises. Proof search may have to make choices, and backtrack over them if needed, as would happen, for instance, if a conclusion of the form $g_1 \vee g_2 \circ [u_1, u_2]$ is encountered, since any of the two rules (R-$\vee_1$) and (R-$\vee_2$) may potentially prove such a conclusion. The reader may observe that despite the need

L-sequents

$$\frac{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta; [] \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]}\text{L-R} \qquad \frac{\Sigma; \Psi; E; \Delta, d \circ [u_1, u_2]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (d \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-clause}$$

$$\frac{\Sigma; \Psi, c; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (c \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-cons}$$

$$\frac{\Sigma; \Psi; E, i; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (i \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-inter}$$

$$\frac{\Sigma; \Psi; E; \Delta; \Xi :: (h_1 \circ [u_1, u_2]) :: (h_2 \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (h_1 \wedge h_2 \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-}\wedge$$

$$\frac{\Sigma; \Psi; E; \Delta; \Xi :: (h_1 \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2'] \qquad \Sigma; \Psi; E; \Delta; \Xi :: (h_2 \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (h_1 \vee h_2 \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-}\vee$$

$$\frac{\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (\top \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-}\top \qquad \frac{}{\Sigma; \Psi; E; \Delta; \Xi :: (\bot \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-}\bot$$

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Delta; \Xi :: (h \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (\exists x{:}\sigma.h \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-}\exists$$

$$\frac{\Sigma; \Psi; E; \Delta, k \text{ claims } d \circ [u_1, u_2]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (k \text{ says } d \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}\text{L-says}$$

$$\frac{\Sigma; \Psi; E; \Delta; \Xi :: (h \circ [u_1', u_2']) \overset{\nu}{\Leftarrow} g \circ [u_1'', u_2'']}{\Sigma; \Psi; E; \Delta; \Xi :: (h \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1'', u_2'']}\text{L-@}$$

Figure 6.3: Goal-directed search, part 2

for making such choices, by eliminating the possibility of applying left rules except after the decomposition of an implication, or after an uninterpreted atom is reached, R-sequents curtail significantly the non-determinism that would be inherent in any proof search tool that naively used the sequent calculus backwards.

Q-sequents. The rules for Q-sequents $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ are fairly straightforward (Figure 6.2). The list $\mathcal{Q}$ is decomposed inductively by the rules (Q-leq) and (Q-goal) depending on the form of query at its head, and the provability of the query is checked either through the constraint solver if the query has the form $u_1 \leq u_2$ (rule (Q-leq)), or through an R-sequent if the query has the form $g \circ [u_1, u_2]$ (rule (Q-goal)). The reader may observe that the rules applying to Q-sequents are deterministic.

F-sequents

$$\frac{}{\Sigma; p \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (u_1 \leq u_1') :: (u_2' \leq u_2) :: []; \mathtt{ff}}\text{F-init}$$

$$\frac{\Sigma; d_1 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}\text{F-}\wedge_1 \qquad \frac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}\text{F-}\wedge_2$$

$$\frac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ \phi) :: \mathcal{Q}; \mathtt{tt}}\text{F-}\supset_1{}^*$$

(${}^*\phi$ **denotes the fixed interval** $[+\infty, -\infty]$**, which is contained in all other intervals**)

$$\frac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ [u_1', u_2']) :: \mathcal{Q}; \mathtt{ff}}\text{F-}\supset_2$$

$$\frac{\Sigma \vdash t : \sigma \qquad \Sigma; d[t/x] \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}{\Sigma; \forall x{:}\sigma.d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}\text{F-}\forall$$

$$\frac{\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}{\Sigma; d \mathbin{@} [u_1'', u_2''] \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}\text{F-}@$$

N-sequents

$$\frac{d \circ [u_1, u_2] \in \Delta \qquad \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}\text{N-clause}$$

$$\frac{\begin{array}{c} k \text{ claims } d \circ [u_1, u_2] \in \Delta \qquad \nu = k_0, u_b, u_e \\ \Sigma; \Psi \models k \succeq k_0 \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \\ \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}\text{N-claims}$$

Figure 6.4: Goal-directed search, part 3

L-sequents. An L-sequent $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ arises during proof search only in the premise of the rule (R-$\supset$). It decomposes the connectives in chunks in Ξ from right to left using the rules of Figure 6.3, which are similar to the corresponding left rules of the sequent calculus. The intuitive justification for completeness of such decomposition with respect to the sequent calculus is that the left rules of the connectives decomposed in L-sequents – $\wedge$, $\vee$, $\exists$, says, and @ – are invertible in the sequent calculus, i.e. their premises hold without principal formulas, whenever their conclusion does (Lemma 6.4).

Due to syntactic restrictions, every chunk eventually decomposes to the forms c, i, d, and k says d. These are pushed into Ψ, E, Δ, and Δ respectively using the rules (L-

cons), (L-inter), (L-clause), and (L-says). Once decomposition of chunks Ξ is complete, which must happen eventually because each backwards application of a rule in Figure 6.3 removes at least one connective from Ξ, the proof search transitions back to an R-sequent and decomposition of the goal is resumed (rule (L-R)). The rules applying to L-sequents are deterministic.

F-sequents. An F-sequent $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ is used to determine whether $d \circ [u_1, u_2]$ can be helpful in proving $p \circ [u_1', u_2']$ or not. During proof search, both $\mathcal{Q}$ and b are outputs. If the sequent has a derivation, then $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$ whenever $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$ (Lemma 6.1). In the following we describe the rules that apply to F-sequents (Figure 6.4), and also explain the role of the boolean b, starting with the latter.

Informal explanation for using the boolean b. An F-sequent $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ is derived by decomposing the connectives in d, and the boolean b is $\mathtt{tt}$ if and only if an @ connective is encountered during this decomposition. Keeping track of whether or not an @ connective is encountered in the decomposition of a clause is important in order to maintain completeness with respect to the sequent calculus. We explain this with a simple example where we try to prove $\Sigma; \Psi; E; \Gamma, g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1'', u_2'']$ in the sequent calculus by decomposing the hypothesis $g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2']$ with left rules. For simplicity, we will assume that $\Sigma; \Psi \models u_1 \leq u_1''$ and $\Sigma; \Psi \models u_2'' \leq u_2$. The reader should note that the scope of the @ connective includes g_2 but not g_1. Using the rule ($\supset$L) once, it suffices to prove the following two sequents for any interval $[u_b, u_e]$ that satisfies $\Sigma; \Psi \models u_1' \leq u_b$ and $\Sigma; \Psi \models u_e \leq u_2'$.

$$\Sigma; \Psi; E; \Gamma, g_1 \supset ((g_2 \supset p) @ [u_1, u_2])) \circ [u_1', u_2'] \xrightarrow{\nu} g_1 \circ [u_b, u_e]$$

$$\Sigma; \Psi; E; \Gamma, g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2'], (g_2 \supset p) @ [u_1, u_2] \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1'', u_2'']$$

Next, using the rule (@L) it suffices to show the following sequent in place of the second sequent above.

$$\Sigma; \Psi; E; \Gamma, g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2'], g_2 \supset p \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1'', u_2'']$$

Observe that due to the connective @, the interval $[u_b, u_e]$ has disappeared from the sequent! Proceeding further, it is easy to show using rules ($\supset$L) and (init) that to establish the third sequent above, it suffices to prove the following sequent.

$$\Sigma; \Psi; E; \Gamma, g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2'], g_2 \supset p \circ [u_1, u_2] \xrightarrow{\nu} g_2 \circ [u_1'', u_2'']$$

Thus in order to prove $p \circ [u_1'', u_2'']$ from the hypothesis $g_1 \supset ((g_2 \supset p) @ [u_1, u_2]) \circ [u_1', u_2']$ assuming that $\Sigma; \Psi \models u_1 \leq u_1''$ and $\Sigma; \Psi \models u_2'' \leq u_2$, it is enough to prove $g_1 \circ [u_b, u_e]$ for *any* $[u_b, u_e]$ that is contained in $[u_1', u_2']$ (first sequent above) and also $g_2 \circ [u_1'', u_2'']$ (fourth sequent above). The relevant observation here is that due to the @ connective inside the head of the implication $g_1 \supset ((g_2 \supset p) @ [u_1, u_2])$, the interval $[u_b, u_e]$ over which g_1 needs to be established has no relation to the interval in the conclusion, $[u_1'', u_2'']$. This observation

generalizes easily – given a hypothesis $g \supset d \circ [u_1, u_2]$, if during the decomposition of d by left rules, an @ connective is encountered, then the interval over which g is proved has no relation to the interval over which the concluding atom produced by d holds. As a result, in this case it is okay in the sequent calculus to prove g over any interval contained in $[u_1, u_2]$. To maintain completeness, when such cases arise in goal-directed search we try to prove g over a fixed interval ϕ that is contained in all other intervals (this interval is defined below).

It should also be noted that if during the decomposition of d by left rules an @ connective is not encountered, then g must be proved over the same interval that is needed in the conclusion derived from $g \supset d$. This happens, for instance, for the clause $g_2 \supset p$ in the third sequent above. The boolean b in the output of F-sequents keeps track of whether an @ connective has been encountered in the decomposition of the clause or not – b is $\mathtt{tt}$ in the sequent $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ if and only if the decomposition of d passes through an @ connective. Accordingly, only the rule (F-@) in Figure 6.4 introduces $\mathtt{tt}$ in its conclusion; all other rules with premises simply propagate the boolean from the premise to the conclusion. Although possibly non-intuitive, this strategy of keeping track of @ connectives encountered during decomposition of clauses is both sound and complete, as the results in §6.3 show. The explanation of the rules for F-sequents below should be easy to follow, given this intuitive understanding of the importance of b.

Explanation of the rules for F-sequents in Figure 6.4. Rule (F-init) means that $p \circ [u_1, u_2]$ can be used to derive $p \circ [u_1', u_2']$ if $u_1 \leq u_1'$ and $u_2' \leq u_2$. This is analogous to the (init) rule in the sequent calculus. Note that the boolean in the output is $\mathtt{ff}$ here. The most interesting rules for deriving F-sequents are (F-$\supset_1$) and (F-$\supset_2$), which are applicable when the booleans in the premise are $\mathtt{tt}$ and $\mathtt{ff}$ respectively. We explain rule (F-$\supset_1$) first. Suppose $\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}$, as in premise of the rule. So $\Sigma; \Psi; E; \Gamma, d_2 \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$ whenever $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$, and further during the decomposition of d_2, an @ connective is encountered. Due to the explanation above, if $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$ and $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} g_1 \circ [u_b, u_e]$ for *any* interval $[u_b, u_e]$, then $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$. Since we cannot represent "any interval" in the logic, we instead choose an interval $\phi = [+\infty, -\infty]$ that is contained in all other intervals. Then the conclusion of the rule $\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ \phi) :: \mathcal{Q}; \mathtt{tt}$ is immediately justified.

The rule (F-$\supset_2$) is similar, but it applies when the boolean in the premise is $\mathtt{ff}$, i.e. when an @ connective is not encountered during the decomposition of d_2. In that case, the goal g_1 must be established on the interval $[u_1', u_2']$, not an arbitrary interval $[u_b, u_e]$ in order for $p \circ [u_1', u_2']$ to be provable. Either of the rules (F-$\wedge_1$) and (F-$\wedge_2$) may apply when the clause to be decomposed has a top level conjunction, so F-sequents may cause backtracking during proof search. At a more fundamental level that we don't explicitly explore here, a derivation of an F-sequent corresponds to a single step of left focusing [12].

N-sequents. N-sequents are the site of backchaining for proving atomic goals of the form $p \circ [u_1', u_2']$. There are only two rules to establish an N-sequent: (N-clause) and (N-claims), both of which are shown in Figure 6.4. For both rules, the objective is to prove $p \circ [u_1', u_2']$

from the assumptions $\Sigma; \Psi; E; \Delta$. In (N-clause) this proof is attempted by choosing a hypothesis of the form $d \circ [u_1, u_2]$ in Δ, finding a $\mathcal{Q}$ such that $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$, and checking that $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$. The rule (N-claims) is similar, except that a hypothesis of the form k claims $d \circ [u_1, u_2]$ is used. In that case, it must also be checked that $k \succeq k_0$, $u_1 \leq u_b$, and $u_e \leq u_2$, where $\nu = k_0, u_b, u_e$. These checks are based on the view principle from §4.2.2. Again, due to the possible choice in picking a suitable hypothesis in Δ, N-sequents may be the site of a significant amount of backtracking during proof search.

6.3 Soundness and Completeness of Proof Search

We now show that goal-directed proof search on the fragment BL_G, as formalized by the rules of §6.2.1, is sound and complete with respect to the sequent calculus. Owing to the presence of explicit time, and particularly because of the need to keep track of @ connectives in clauses during decomposition in F-sequents, proofs of both soundness and completeness are somewhat more difficult than they are for other logics. Details of these proofs are provided in Appendix D, and we encourage the interested reader to go through the appendix to gain an insight into the technical details of the proofs. Here we only skim through the main lemmas.

Soundness. For soundness, we seek to establish that if $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$, then $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g \circ [u_1, u_2]$. The proof of this fact critically relies on the following lemma about soundness of F-sequents.

Lemma 6.1 (Soundness of F-sequents)**.** *Suppose the following hold.*

1. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$
2. $\Sigma; \Psi; E; \Gamma \overset{\nu}{\rightarrow} \mathcal{Q}$

Then, $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \overset{\nu}{\rightarrow} p \circ [u_1', u_2']$.

Proof. We generalize the statement of the lemma, one for each of the two cases $b = \mathtt{tt}$ and $b = \mathtt{ff}$, which can then be proved using simultaneous induction on the derivation assumed in (1). See Appendix D, Lemmas D.2 and D.3 for details. □

Using this lemma, soundness follows by an easy induction on derivations of goal-directed search.

Theorem 6.2 (Soundness)**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, |\Xi| \overset{\nu}{\rightarrow} g \circ [u_1, u_2]$

C. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} p \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} \mathcal{Q}$

Proof. By simultaneous induction on given derivations and case analysis of the last rules in them. For proving (C), Lemma 6.1 is needed. Representative cases may be found in Appendix D, Theorem D.4. □

Completeness. For completeness, we seek to establish that if $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g \circ [u_1, u_2]$, then $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$. To prove this fact, we show first that the left rules for connectives that appear in clauses d are admissible in goal-directed search. This requires a systematic development of the metatheory of goal-directed search which we defer to Appendix D, as well as several tedious but straightforward inductions over derivations of goal-directed search. Second, we prove that the left rules for top level connectives in chunks are all invertible in the sequent calculus. This is straightforward. Given these two lemmas, completeness follows by a lexicographic induction on sequent calculus derivations.

Lemma 6.3 (Admissibility of left rules in proof search)**.** *The following hold.*

1. *Admissibility of* ($\supset L$)*. Suppose the following hold:*

   (a) $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$

   (b) $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} g'' \circ [u_1'', u_2'']$

   (c) $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$

   Then $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g'' \circ [u_1'', u_2'']$

2. *Admissibility of (claims). Suppose the following hold:*

   (a) $\nu = k_0, u_b, u_e$

   (b) $\Sigma; \Psi \models k_0' \succeq k_0$

   (c) $\Sigma; \Psi \models u_b' \leq u_b$

   (d) $\Sigma; \Psi \models u_e \leq u_e'$

   (e) $\Sigma; \Psi; E; \Delta, k_0' \ \mathsf{claims}\ d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

   Then, $\Sigma; \Psi; E; \Delta, k_0' \ \mathsf{claims}\ d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

3. *Admissibility of* ($\wedge L$)*. Suppose the following holds:*

   (a) $\Sigma; \Psi; E; \Delta, d_0 \wedge d_0' \circ [u_b', u_e'], d_0 \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

   Then, $\Sigma; \Psi; E; \Delta, d_0 \wedge d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

4. *Admissibility of* ($\forall L$)*. Suppose the following hold:*

   (a) $\Sigma \vdash t : \sigma$

   (b) $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma. d_0 \circ [u_b, u_e], d_0[t/x] \circ [u_b, u_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

Then, $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e] \stackrel{\nu}{\Rightarrow} g \circ [u_1, u_2]$

5. *Admissibility of (@L). Suppose the following holds:*

   (a) $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e], d_0 \circ [u'_b, u'_e] \stackrel{\nu}{\Rightarrow} g \circ [u_1, u_2]$

   Then, $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e] \stackrel{\nu}{\Rightarrow} g \circ [u_1, u_2]$

Proof. Each of the statements (1)–(5) is proved separately by induction on the last given R-sequent derivation. In each case, the induction hypothesis must be generalized to include L-sequents, Q-sequents, and N-sequents. Details are presented in Appendix D, Lemmas D.11–D.15. The proof of statement (1) is particularly involved and requires a systematic development of metatheory of goal-directed search, including an analogue of time subsumption (Theorem 4.11). We strongly encourage the interested reader to look at Appendix D for all the details. □

Lemma 6.4 (Strong left inversion for $c, i, \wedge, \vee, \top, \exists, \mathsf{says}, @$)**.** *The following hold for the sequent calculus of BL.*

1. $\Sigma; \Psi; E; \Gamma, c \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi, c; E; \Gamma \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

2. $\Sigma; \Psi; E; \Gamma, i \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi; E, i; \Gamma \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

3. $\Sigma; \Psi; E; \Gamma, s_1 \wedge s_2 \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2], s_2 \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

4. $\Sigma; \Psi; E; \Gamma, s_1 \vee s_2 \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies both* $\Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *and* $\Sigma; \Psi; E; \Gamma, s_2 \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by derivations of smaller or equal depth.*

5. $\Sigma; \Psi; E; \Gamma, \top \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi; E; \Gamma \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

6. $\Sigma; \Psi; E; \Gamma, \exists x{:}\sigma.s \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma, x{:}\sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

7. $\Sigma; \Psi; E; \Gamma, k \mathsf{\ says\ } s \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi; E; \Gamma, k \mathsf{\ claims\ } s \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

8. $\Sigma; \Psi; E; \Gamma, s \mathbin{@} [u''_1, u''_2] \circ [u_1, u_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *implies* $\Sigma; \Psi; E; \Gamma, s \circ [u''_1, u''_2] \stackrel{\nu}{\longrightarrow} r \circ [u'_1, u'_2]$ *by a derivation of smaller or equal depth.*

Proof. Each statement follows by a separate induction on the depth of the given derivation and a case analysis of the last rule in the derivation. See Appendix D, Lemma D.18 for some representative cases. □

Theorem 6.5 (Completeness)**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_0 \circ [u_0, u_0']$

B. $\Sigma; \Psi; E; \Delta, |\Xi| \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ *implies* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$

Proof. By simultaneous lexicographic induction, first on the depths of the given derivations, and then on the order (B) > (A). For (B), we also subinduct on the number of connectives in Ξ. Details are in Appendix D, Theorem D.19. Briefly, to prove (A), we case analyze the last rule in the derivation of $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_0 \circ [u_0, u_0']$. If it is a right rule, we apply the i.h. to the premises and use the corresponding rule for R-sequents (Figure 6.2). If it is a left rule, we apply the i.h. to the premises and use the corresponding clause from Lemma 6.3. To prove (B), we subinduct on the number of connectives in Ξ, and analyze the form of the last chunk in it. Then we use an appropriate clause of Lemma 6.4 to reduce the problem to one with a Ξ with fewer connectives, apply the i.h., and then apply the corresponding rule for L-sequents from Figure 6.3. □

6.4 Implementation in PCFS (and Otherwise)

Whereas it should be clear that by preventing arbitrary interleaving of left and right rules, goal-directed search reduces non-determinism inherent in the sequent calculus, there are two points that probably need explanation before it will be clear how the rules of §6.2.1 can be efficiently implemented.

- How does proof search resolve the choices in choosing between the rules (R-$\vee_1$) and (R-$\vee_2$) on one hand, and (F-$\wedge_1$) and (F-$\wedge_2$) on the other, as well as the choice in picking a hypothesis in Δ to apply one of the rules (N-clause) and (N-claims)?

- How does proof search "guess" the terms t in the premises of the rules (R-$\exists$) and (L-$\forall$)?

The answer to the first question is that, in order to remain complete with respect to the sequent calculus, the proof search tool must explore all these choices. This may be done breadth-first to obtain a semi-decision procedure for BL_G, but in practice it is much better to use depth-first search with backtracking, which is also what the PCFS proof search tool `pcfs-search` does. Like all logic programming languages, it is possible to optimize choices in N-sequents by deriving all possible F-sequents from given clauses Δ in advance of their actual use in N-sequents. This form of clause-compilation is called *residuation* in logic programming (see, e.g., [38]). Its application to goal-directed search in BL_G requires a slight modification to F-sequents and their rules, since the goal $p \circ [u_1', u_2']$ will not be available at the time that residuation is performed. Although residuation is useful in improving efficiency of goal-directed search, the current implementation of the PCFS tool `pcfs-search` does not perform this optimization.

The second problem above, that of picking the term t in the premises of the rules (R-$\exists$) and (L-$\forall$), may be resolved in an implementation using a standard approach based on unification. Instead of guessing a term t, proof search inserts a new variable, called an existential variable or evar, in its place. This variable is then resolved (its substitution found)

during the rule (F-init) by a process of unification. This is completely standard in logic programming as well as theorem proving, and requires little further explanation. The only noteworthy aspect in BL_G, as also in other logic programming languages or theorem provers with constraint domains, is that there is inherently a possibility of requiring unification in the constraint solver. More precisely, what happens if we invoke the constraint solver with a judgment of the form $\Sigma; \Psi \models c$ which contains evars? In general, for many forms of constraints, it is possible to unify evars (or find possible satisfying ranges for integer evars) within the constraint domain.

In the context of authorization policies, unification within the constraint solver may be unnecessary in practice – at least for all authorization policies in this thesis, it is the case that most invocations of the constraint solver during goal-directed proof search are with ground constraints. Therefore, the PCFS proof search tool `pcfs-search` assumes in most cases that all terms during any invocation of the constraint solver are ground. The only exception to this rule is the constraint $\texttt{is}\ t\ t'$, which checks equality of t and t', modulo rules of arithmetic. In this case, following usual logic programming conventions, `pcfs-search` requires that t not contain any arithmetic operators, and that t' be a ground arithmetic expression over time points. It simplifies t' using arithmetic rules, and unifies the result with t. Generally, the requirements that constraints be ground entails at the least that all terms in the view ν of a sequent, as well as annotations of the form $\cdot \circ [u_1, u_2]$ and $k\ \mathsf{claims}\ \cdot$ remain ground during proof search. Whether this will be the case for a policy or not may be checked with a simple mode analysis over the formalized policy rules (e.g., as implemented in Twelf [116]).

A similar groundedness assumption on interpreted predicates is unrealistic. For example, during proof search over policy rules of §8, an interpreted predicate of the form **owner** $f\ K$ must often be resolved, where K is unknown. So the solver for interpreted predicates in PCFS does perform unification, but only in certain arguments. For example, for the predicate **owner** $f\ K$, it requires that f be ground, else it would not know which file's meta-data to look at. Again, a static mode analysis on authorization policy rules may be used to check that such requirements will be satisfied at all times during proof search. Since a proof may be constructed in a system state different from the one in which the authorization derived from it will be used, `pcfs-search` can be run in interactive mode, where it asks the user about the truth of every interpreted predicate that it deems useful to the proof, instead of checking the predicate on the system state.

The actual implementation of the tool `pcfs-search` uses the rules of §6.2.1 with changes described above. The implementation is written in Standard ML, and is based on an earlier implementation of goal-directed search for intuitionistic logic by Pfenning and Elliott using success continuations, imperative unification, and implicit backtracking through native sequencing of programs in the programming language [57]. Even without any optimizations, the implementation is reasonably fast. Proofs from policies of §8, which often contain well over 1000 inference rules and refer to over 70 policy rules, can be constructed in less than 300ms by the tool, including the time for reading and parsing certificates from disk.

6.5 Related Work

The formalization of goal-directed search in §6.2 is based on a similar formalization for inference in the linear logic programming language Lolli [78], and indirectly on Miller et al.'s work on uniform proofs [103]. To our best knowledge, the latter was the first piece of work to explicitly relate goal-directed search via inference rules to logic programming languages, an approach on which the theory and implementation of `pcfs-search` builds. Goal-directed search is an instance of a more general method of restricting the search space for proofs without losing completeness, called focusing, that was introduced by Andreoli in the context of classical linear logic [12]. The method has been adopted in other logics, including intuitionistic logic [80, 101], and intuitionistic linear logic [43]. Very recently, Licata and Morgenstern have built a theorem prover for all of BL_S using focusing (personal communication). The work in this chapter, although influenced by a lot of prior work, is unique in the sense that it considers the hybrid modality s @ $[u_1, u_2]$, which as explained earlier presents a number of unique challenges both in the design of goal-directed proof search and in the proofs of its soundness and completeness.

Within the context of authorization, a significant amount of work on proof search has focused on the problem of storing and finding the hypotheses needed to construct a proof. This problem is generically called credential chain discovery, and was first studied by Clarke et al. for SPKI [46]. Its scope was greatly expanded in the trust management framework RT [97], and in the work of Bauer et al. [21] whose experimental set up was Grey, a proof-carrying authorization framework. The problem of credential chain discovery is orthogonal to the concerns of this chapter since we are interested in finding proofs assuming that relevant hypotheses are available (we allow proof search to fail if they are not), whereas most work on credential chain discovery focuses on finding hypotheses, and generally simplifies the problem of constructing a logical proof to the use of heuristics. The difference in the two approaches may be explained partly by a difference in setting – most work on credential chain discovery is based in settings where certificates are distributed on a network, and finding them is the difficult part, but policies themselves are simple. On the other hand we are more interested in the problem of finding proofs from complex policies, where general heuristics may be hard to describe. Recently, Becker et al. [24] have studied the problem of credential chain discovery using abduction in logic programming, which is interesting because it combines these two rather orthogonal problems in a single framework.

Most logic-based languages for authorization policies include some procedure for automatic inference from policy rules. Many of these, e.g., [23, 52], translate policies and goals to Datalog and use saturating search to perform inference. Others like Soutei use backchaining search [118]. Saturating search works well when all consequences of a policy are needed, for instance, to compile the policy to low-level configurations. However, goal-directed search may be more efficient for constructing proofs of individual authorizations, which is the case in PCFS.

Chapter 7

The Proof-Carrying File System (PCFS)

This chapter presents details of the design and implementation of PCFS, as well as experimental measurements that evaluate its performance during file access. A high-level picture of the PCFS architecture was presented in §2, details of the use of our authorization logic BL in PCFS were the subject of §4.3, the PCFS proof verification procedure was described and formalized in §5.2, and a method for automatic proof search for PCFS was presented in §6. This chapter explains how all these fit together, and more significantly, it explains the layout of files and configuration information within a PCFS file system, and how PCFS uses procaps (capabilities) generated from proofs to authorize file operations during file access.

As mentioned in §2, the PCFS architecture is divided into two parts, both conceptually and in the implementation: (a) The front end that deals with policies, builds on BL's proof theory to authorize access based on proofs, and generates procaps, and (b) The back end that handles file system calls, and uses procaps to authorize file operations. This chapter is organized around these two parts. §7.1 describes the command line tools for managing policies and proofs that comprise the front end, whereas §7.2 explains the layout of the file system, how system calls are intercepted and, the use of procaps to authorize access. §7.3 presents the results of benchmarks which establish that procap-based authorization checks are efficient in practice. §7.4 summarizes assumptions made by PCFS about its operating environment and also describes the trusted code base of its implementation.

Since work related to PCFS was already presented at the end of §2, related work is not included in this chapter. Some of the content of this chapter first appeared in a technical report authored jointly with Pfenning [68].

7.1 The PCFS Front End

The content of this section relies on that of §2.1, §4.3, and §5.2. Readers may wish to revisit those sections before continuing.

The front end of PCFS consists of command line tools that create and manage digital

certificates, automatically search for proofs, verify proofs to generate procaps, and put procaps in the procap store. Since all these tools are used prior to file access, and all of them are invoked rarely in comparison to the frequency of file access itself, efficiency of the tools is not a significant concern in PCFS. In contrast, the back end needs to be, and is, very efficient. In the following we describe each of these tools. They are all written in Standard ML, and together comprise approximately 7,000 lines of code. OpenSSL is used for all cryptographic operations [2]. RSA keys are used for signing and verifying digital certificates, while procaps are signed using HMACs made with a symmetric key. Details of both RSA and HMACs can be found in any text on applied cryptography, e.g., [102].

Certificates and keys. PCFS uses its own XML-based format for digital certificates, which are of two kinds. The first kind of certificates, called policy certificates, establish policy rules. Abstractly, a policy certificate is a five-tuple $\langle k, u_1, u_2, s, sig\rangle$, where k is the creator of the certificate, $[u_1, u_2]$ is the interval of validity of the certificate, s is the policy formula asserted, and sig is a digital signature over the other four elements generated using k's signing (private) key. As mentioned in §4.3.1, such a certificate is represented in BL as the judgment k claims $s \circ [u_1, u_2]$.

The second kind of certificates, called key certificates, map principals to their verification (public) keys. We assume that such certificates are signed by a fixed principal called the certifying authority (CA). The exact name or user id of this principal is irrelevant. What is important is its public key that can be used to check such certificates. This key is stored in a specially protected file called `ca-pubkey.pem` within the PCFS file system (see §7.2.2). Abstractly, a key certificate is a triple $\langle k, key, sig\rangle$, where k is the principal whose key it certifies, key is the public key being certified, and sig is a signature on the first two components generated with the CA's signing key.

PCFS provides a command line tool called `pcfs-cert` which can be used to create certificates of both kinds and to verify sets of certificates. For creating a policy certificate the tool expects a private key that is used to sign the certificate, whereas for creating a key certificate it expects the CA's private key. A set of certificates is verified by checking the signature on each key certificate using the CA's public key, and checking the signature on each policy certificate using the public key of the principal mentioned in the certificate; the latter key is obtained from one of the key certificates.

Proof search. PCFS includes an automatic proof search tool called `pcfs-search`, described in §6, that helps users find proof terms V such that $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ (see §5.1 for an explanation of this judgment). The user is responsible for providing Π in the form of policy certificates, s which should usually have the form admin says (`may` k f η), u_1, and u_2. The tool reads Σ from a file `declarations` stored within the PCFS file system (see §7.2.2), and by default assumes that E is the prevailing system state. The latter behavior may be changed through a command line option, which causes the tool to prompt the user about every interpreted atom that may be useful in the proof. This may be necessary if the proof is being generated in a system state that is different from the one in which the procap obtained from it is expected to be used.

The proof construction tool is not a trusted component of PCFS, nor are users obliged to use it. Instead, users may construct proofs by any means they like. However, given that proofs of authorization can be quite complex, as happens for instance with the policies of §8, a fast, automatic tool to find proofs is quite useful in practice. `pcfs-search` is based on a logic programming interpretation of a large, expressive fragment of BL and works quite fast. Typical proofs from policies of §8 requiring as many as 1100 inference steps can be constructed by the tool in less than 300ms.

Proof verification. Proofs establishing access must be verified using a PCFS command line tool called `pcfs-verify`. Unlike the proof search tool, this tool is a trusted component of PCFS, and users are obliged to use it. The tool implements the verification algorithm explained in §5.2 and relies on three special files `config-file`, `declarations`, and `shared-key` that are within the PCFS file system and protected by it (see §7.2.2 for details). Specifically, given a proof term V, policy rules Π in the form of certificates, k, f, and η, the tool proceeds as follows.

- It checks all certificates as explained earlier.

- It reads the sorting Σ from the file `declarations`. This file also contains the arities and expected sorts of arguments of all predicate and function symbols, using which the well-formedness of all formulas in Π, and the well-formedness of s, k, f, η, and V are checked.[1]

- It reads the identity of principal admin who is responsible for authorizing access from `config-file`.

- It tries to find $\mathcal{C}$ and $\mathcal{I}$ such that $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow \mathsf{admin\ says}\ (\mathtt{may}\ k\ f\ \eta) \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$, as explained in §5.2.1.

If all checks succeed the tool reads the key for signing procaps from the file `shared-key`, and issues the procap $\langle k, f, \eta, \mathcal{C}, \mathcal{I}, sig\rangle$ to the user. As explained in §5.2.3, this procap can be used by k to authorize permission η on file f whenever the conditions $\mathcal{C}$ and $\mathcal{I}$ are satisfied. In particular, if V satisfies $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow \mathsf{admin\ says}\ (\mathtt{may}\ k\ f\ \eta) \circ [u_1, u_2]$, then as discussed in §5.2.2, these conditions will always be satisfied in system state E, provided that the time of access lies between u_1 and u_2.

The tool `pcfs-verify` must run with the user id of a privileged user called pcfssystem in order to read the secret key for signing procaps, as discussed in greater detail in §7.2.2. This is ensured by making pcfssystem the owner of the binary file `pcfs-verify`, and setting the setuid bit on it.

[1] We have not formalized in this thesis how well-formedness is checked, but it is straightforward. Briefly, well-formedness ascertains that the arguments of all predicates and function symbols have the stipulated sorts.

Procap injection. Communication of procaps from the front end to the back end of PCFS is via a store within the PCFS file system. The front end provides a command line tool called `pcfs-qprocap` to inject procaps generated by the verifier into this store.[2] Alternatively, the beneficiary of a procap may copy the procap into the store directly. The organization of the procap store is described in §7.2.2.

7.2 The PCFS Back End

The PCFS back end handles file system calls, looks up relevant procaps in the procap store, checks them to authorize access, and performs file I/O. In this sense the back end of PCFS is the actual "file system"; the front end supports it by providing a BL-based framework for generating procaps from proofs of authorization. The back end can be used as a file system with any other mechanism that can generate procaps.

The current implementation of the PCFS back end is a local file system for Linux-based operating systems that contain the Fuse kernel module [1]. Technically, PCFS is a *virtual file system* that performs procap-based authorization checks and then uses an underlying file system for disk I/O. For all experiments reported in this chapter, the underlying file system is ext3. Using an underlying file system has the merit that it avoids the need to re-implement disk I/O, but the two-tier architecture adversely affects performance, particularly because it involves inter-process communication (explained below). Even with this virtual organization the performance of the back end is very good and high enough for most practical intents and purposes (see §7.3 for details).

PCFS is mounted using the command, where `/path/to/src` is an existing directory in an ext3 (or other) file system, and `/path/to/mountpoint` is an empty directory.

```
$> sudo pcfs-main /path/to/src /path/to/mountpoint
```

After the execution of this command, any file system call on a path like `/path/to/mountpoint/foo/bar` results in a corresponding operation on `/path/to/src/foo/bar`, but is subject to procap-based authorization checks that are described in §7.2.1. To prevent users from directly using the underlying file system to access data, it is expected that ownership of `/path/to/src` on the underlying file system will be set to the superuser, and all access on it will be turned off.[3]

The directory `/path/to/src` must contain a special subdirectory named `/path/to/src/#config`, which is visible in the mounted PCFS file system at `/path/to/mountpoint/#config`. This subdirectory contains the procap store as well as other configuration parameters such as the symmetric key used to sign and verify procaps, the public key of the

[2] The letter 'q' in the name `pcfs-qprocap` stands for 'quantified', which is an allusion to the ability of the tool to substitute free term variables in a procap with ground terms before injecting the resulting procap into the procap store. Although such substitutions are not useful in ordinary practice, they are helpful if procaps are generated automatically by a compiler. Details of the latter may be found in joint work of the author and Chaudhuri [41].

[3] A more secure method to prevent access via the underlying file system is to keep data encrypted on it, and to decrypt data in the PCFS back end after making access checks. We have not implemented this design, since our objective here is to evaluate the performance of access checks.

certifying authority (§7.1), arities and expected sorts of arguments of all predicates and function symbols, and the user id of the principal `admin`. Since `/path/to/mountpoint/#config` contains sensitive information that controls access to all other files and directories, access to `/path/to/mountpoint/#config` and its subpaths is not governed by procaps, but by rules that are hardcoded in the back end. The organization of this special subdirectory and rules for access to it are discussed in §7.2.2. Access to all files and directories outside `#config` is subject to procap-based checks.

Before proceeding to explain the back end in detail, we would like to summarize its implementation. The Fuse kernel module on which PCFS builds works by trapping any file system calls made on subpaths of `/path/to/mountpoint` and redirecting them to a user-level server process, which the developer of the file system must provide. The interaction between the kernel and the user-level server process occurs over an inter-process communication (IPC) channel. The Fuse development tool kit also provides a stub for developing the server process, reducing the code development effort to writing one function for handling each file system call like open, read, write, stat, unlink, rmdir, mkdir, etc. In PCFS, these handling functions look up relevant procaps from the procap store, parse them, check them, decide whether or not to allow access, and then repeat the same call that they were handling on `/path/to/src`, after which the underlying file system performs I/O. (The server process runs as superuser so it is not subject to any access checks in the underlying file system.) It is in handling the procaps that most of the development effort in the back end lies. In addition to looking up, parsing, and checking procaps, the back end also includes an in-memory cache that stores frequently used procaps in parsed form. This cache is described in §7.2.1. The entire implementation of the back end contains approximately 10,000 lines of C++ code, and relies on the OpenSSL library for verifying the cryptographic signature on procaps.

Backwards compatibility. PCFS is highly backwards compatible. Most programs like word processors, spreadsheets, shell commands, compilers, and file utilities including automatic file indexers work on it without problems. Part of this compatibility is implicit in the fact that the use of Fuse ensures that PCFS exposes the standard Linux system call API to programs. However, this alone does not suffice. Two other design decisions complement the use of Fuse in providing backwards compatibility:

- First, the use of a procap store, accessible to both users and the back end ensures that no extra arguments are needed to pass procaps in file system calls.

- Second, files created by programs remain accessible to them temporarily even in the absence of policy rules due to procaps generated by the file system itself. This is explained in §7.2.1.

However, PCFS is not completely backwards compatible because it does not strictly follow POSIX specifications for access checks. As explained in §7.2.1, deviation from POSIX in this regard is deliberate, and increases the range of policies that can be enforced with PCFS.

Operation	Permissions needed
stat `/foo`	execute on `/foo`
open `/foo` in read mode	read on `/foo`
open `/foo` in write mode	write on `/foo`
open `/foo` in read/write mode	read and write on `/foo`
opendir `/bar`	read on `/bar`
create `/bar/foo`	write on `/bar`
delete `/bar/foo`	identity on `/bar/foo`
rename `/bar` to `/foo1/foo2`	identity on `/bar`, write on `/foo2` if `/foo1/foo2` exists identity on `/bar`, write on `/foo1` if `/foo1/foo2` does not exist
getxattr on `/foo`	execute on `/foo`
setxattr on `/foo`	govern on `/foo` if attribute starts with `user.#pcfs.`, write otherwise
chown on `/foo`	govern on `/foo`
chmod on `/foo`	write on `/foo`

Figure 7.1: Permissions needed to perform common operations in PCFS

7.2.1 Permissions and Access to Files

The content of this section assumes knowledge of §5.2.3. Readers may wish to revisit that section before continuing.

Access to files and directories in the PCFS back end is based on permissions that are authorized through procaps. As explained in §5.2.3, each procap gives a single principal a specific permission on one file or directory. There are five possible permissions – read, write, execute, identity, and govern. When a system call is made to a PCFS file system, the server process first determines the user id of the process making the call. This information is provided by a Fuse interface, through a function similar to the POSIX method `getuid`(). Next, based on this user id, the path name(s) being accessed, and the specific operation being performed, the server process looks up the procap store to find procaps to authorize relevant permissions. The permissions that must be authorized for each operation are listed in Figure 7.1. (Path names listed in the figure are relative to the PCFS mount point.) The procap store is indexed by user ids, path names, and permissions; its organization is described in §7.2.2. If all required procaps are found, they are parsed, their cryptographic signatures are verified, and their conditions are checked as described in §5.2.3. Once all these checks succeed, the operation is performed using the underlying file system. If any steps fail, an access error (POSIX error code EACCES) is returned.

The PCFS permissions execute, read, and write roughly correspond to POSIX permissions of the same names. The execute permission is needed to read meta-data of a file or directory (operations stat and getxattr). The read and write permissions are needed to read and write the contents of a file respectively (operation open). In the case of directories, the read permission allows reading the list of objects in the directory (operation opendir),

while the write permission allows creation of new files in the directory (operation create). In compliance with POSIX specifications, PCFS does not make any access checks during the file system calls read and write. Instead, relevant checks are made when the file is opened for read or write or both. However, PCFS can be forced to check for read and write permissions during every read and write through an option in the PCFS configuration file `config-file` that is explained in §7.2.2.

In addition to the three POSIX permissions execute, read, and write, PCFS uses two more permissions – identity and govern – to allow finer access control. The identity permission is needed for operations delete and rename that change the identity of a file or directory. In contrast from POSIX, we separate the permission to create objects in a directory (write) from the permission to rename or delete an object (identity) because deletion and renaming have effects very different from file creation – deleting or renaming shared files and directories may adversely affect other users' work, and renaming files and directories also affects what policy rules apply to them (recall from §4.3 that files and directories are identified in policy rules using pathnames). An example of the usefulness of the separation between write and identity arises in the case study of §8, where individuals may be allowed to create files with sensitive information but not to delete or rename them.

The govern permission is needed to change meta-data of a file on which policy rules may depend. Recall from §4.3 that the implementation of BL in PCFS supports two interpreted predicates – **owner** f k which checks that the owner of file f is k, and **has_xattr** f a v which checks that extended attribute `user.#pcfs.`a on file f is set to v. Accordingly, to change either the owner (operation chown) or the value of an extended attribute with prefix `user.#pcfs.` (operation setxattr), the permission govern is necessary. Use of the govern permission is illustrated in the example of §4.3.3 and in the case study of §8.

The nine POSIX permission bits, commonly written rwxrwxrwx, have no effect in PCFS, so the write permission suffices to change them (operation chmod). Another significant difference from POSIX specifications is that POSIX requires the execute permission on all ancestors of the file or directory on which an operation is to be performed, but PCFS does not mandate this check. Where necessary, the check may forced by building it in the policy rules. To facilitate the latter, the PCFS implementation includes the constraints `isroot` d and `isparent` d f, which mean respectively that d is the root of the PCFS file system, and that d is the parent directory of f. Using these constraints, it is easy to encode recursive checks on ancestor directories in BL.

Default permissions. Many programs create files during their execution, to which they must have access in order to complete their tasks. To maintain backwards compatibility with such programs, i.e. to not force modifications to the programs in order to generate procaps for access to newly created files, when a new file or directory is created, the PCFS back end automatically creates and injects *default procaps* that give the creator of the file or directory read, write, execute, and identity permissions for a fixed period of time (in the current implementation this period is 90 days, but that can be changed easily). After this period elapses, the default procaps expire and policy rules must be created to control access to the file. Also, every default procap is conditional on a specific extended attribute

`user.#pcfs.newfile` being set to 1. This attribute is set automatically at the time of creation. Anyone with govern permission on the file or directory may prematurely terminate access through default procaps by changing or deleting this attribute. In situations where default procaps are unnecessary, their generation may be suppressed using an option in the configuration file `config-file` as explained in §7.2.2.

7.2.2 Configuration Files and the Procap Store

PCFS relies on a significant amount of configuration information as well as the procap store, both of which must be protected from unauthorized access. As mentioned in the beginning of §7.2, all this information is stored in a directory named `#config` which is present in the root of the PCFS file system. For illustration, we assume throughout this section that PCFS is mounted at `/pcfs`. Accordingly, the configuration directory is `/pcfs/#config`. Access to the configuration directory is not determined through procaps, but via special rules that are hardcoded in the implementation of the back end. In particular, PCFS assumes a special user referred to by the name pcfssystem in this chapter, which has complete access to this directory. This user is expected to perform maintenance tasks on the file system such as changing its configuration files, changing the symmetric key used to sign procaps, or deleting unnecessary procaps. Another important role of pcfssystem is that the proof verifier runs with its user id, since it needs access to the symmetric key to sign procaps. It should be noted that pcfssystem may be (and usually should be) distinct from admin, who controls access via the policy. In fact, pcfssystem should not appear in policies at all.

The following is a list of files and directories within `/pcfs/#config`, together with a description of their contents, and the rules for access to them.

`/pcfs/#config/config-file`: File containing general configuration options, including those listed below. Anyone may read this file, but only pcfssystem may write to it.

1. User ids of the principals admin and pcfssystem.
2. Whether or not default procaps discussed in §7.2.1 are to be generated.
3. Whether or not procaps are to be deleted when the file they authorize is deleted or renamed (explained later in this section).
4. Size of the in-memory procap cache, which is explained later in this section.

`/pcfs/#config/shared-key`: Contains the shared key used to sign procaps. Only pcfssystem may read or write this file.

`/pcfs/#config/ca-pubkey.pem`: Contains the public key of the certifying authority who signs associations between other public keys and users (§7.1). Anyone may read this file, but only pcfssystem may write to it.

`/pcfs/#config/declarations`: Contains the sorts of all constants (i.e. Σ), as well as the arities and expected sorts of arguments of all predicates and function symbols allowed in policies. Anyone may read this file, but only pcfssystem may write to it.

`/pcfs/#config/procaps/`: This directory contains procaps and constitutes what we have been calling the procap store. Its organization is discussed next. pcfssystem has full access to this directory, and other users have access to specific subdirectories only.

The procap giving the right $\langle k, f, \eta \rangle$ is stored in the file `/pcfs/#config/procaps/<`k`>/<`f`>.perm.<`η`>`. Here `<`k`>` is the user id of the user k, `<`f`>` is the path of the file f (relative to the mount point), and `<`η`>` is a textual representation of the permission. Thus each procap is stored in a separate file, and further for each right $\langle k, f, \eta \rangle$, there can be at most one procap that authorizes the right. While this may be restrictive, it makes look up extremely easy since the exact path where a procap is to be found can be determined simply by knowing the PCFS mount point and the right $\langle k, f, \eta \rangle$. To prevent denial of service attacks and to protect user privacy, the PCFS back end ensures that only user k can access (read, write, or delete) files inside `/pcfs/#config/procaps/<`k`>/`.

Since pcfssystem has full access to all files and directories within `/pcfs/#config/`, its user account is a very attractive target for attack. If an attacker gains control of this user account, it can read the secret key used to sign and verify procaps, and inject fake procaps to access other files. To prevent this, the PCFS server process denies pcfssystem all permissions in other directories within the file system.

Automatic procap deletion. To prevent unnecessary procaps from accumulating in the procap store, the PCFS back end, by default, deletes all procaps associated with a file or directory when the latter is deleted or renamed. This is a costly operation, as is evident from microbenchmarks (§7.3). Such automatic deletion of procaps can be prevented by setting an option in the file `/pcfs/#config/config-file`. In its place, pcfssystem may periodically run a simple script that removes all procaps which authorize files that do not exist.

In-memory procap cache. Since procaps are stored in files, and one or more of them must be read to authorize almost every operation on a PCFS file system, it is helpful to cache commonly used procaps in memory to improve performance. Accordingly, PCFS uses a least recently used (LRU) in-memory cache, whose size can be adjusted through an option in the file `/pcfs/#config/config-file`. The cache stores parsed procaps, whose signatures have already been verified. The only cost involved in using a cached procap is checking its conditions ($\mathcal{C}$ and $\mathcal{I}$ from §5.2.1). This is extremely fast and usually takes only 10–100μs. In contrast, seeking the procap on disk may take a few milliseconds, and parsing it often takes up to 70μs. As a result of this cache PCFS obtains extremely high performance when the number of files in use is small. We evaluate the effect of the cache with different hit rates in §7.3. The PCFS back end automatically marks a cached procap dirty if its corresponding file on disk changes or is deleted. This forces the cached procap to be read again from the disk whenever it is needed next.

7.3 Performance Evaluation of the Back End

In this section, we present results of evaluation of the performance of the PCFS back end through benchmarks. Specifically, we evaluate the overhead of access checks during read, write, stat, create, and delete operations, and measure the effectiveness of the in-memory procap cache through microbenchmarks. To evaluate performance in practice, we also present the results of two simple macrobenchmarks. Since we are primarily interested in measuring the overhead of procap-based access checks, our baseline for comparing performance is a Fuse-based file system that does not perform the corresponding checks, but otherwise runs a server process and uses an underlying ext3 file system, just as PCFS does. We call this file system Fuse/Null. For macrobenchmarks we also compare with a native ext3 file system. All measurements reported here were made on a 2.4GHz Core Duo 2 machine with 3GB RAM and a 7200RPM 100GB hard disk drive, running the Linux kernel 2.6.24-23.

Read and write throughput. As mentioned in §7.2.1, by default, PCFS does not make any access checks when read or write operations are performed on a previously opened file. As a result its read and write throughput is very close to that of Fuse/Null. The following table summarizes the read and write throughputs of PCFS and Fuse/Null based on reading and writing a 1GB file sequentially using the Bonnie++ test suite [47].

Operation	PCFS (MB/s)	Fuse/Null (MB/s)
Read	538.69	567.47
Write	73.18	76.05

Even if access checks on every read and write are enabled, the read and write throughputs do not show a significant change as long as required procaps remain cached in memory.

File stats and effectiveness of caching. Besides read and write, two other very common file operations are open and stat (reading a file's meta-data). In terms of access checks, both are similar, since usually one procap must be checked in each case.[4] We report in the table below the speed of the stat operation and the effect of the in-memory procap cache with different hit rates. All measurements are reported in number of operations per second, as well as time taken per operation. The title $n\%$ indicates a measurement with a cache hit rate of $n\%$. For comparison performance of Fuse/Null is also shown. The figures are based on choosing a random file 20,000 times in a directory containing exactly 20,000 files, and stating it. To get a hit rate of $n\%$, the cache size is set to $n/100 \times 20000$, and the cache is warmed a priori with random procaps. It is easy to prove that for an LRU cache this results in a hit rate of exactly $n\%$ when subsequent files (procaps) are also chosen at random. All procaps used here are default procaps, whose conditions include two constraints of the form $u_1 \leq u_2$ (without hypotheses), and one interpreted predicate of each of the two forms **has_xattr** and **owner**.

[4] Two procaps must be checked when a file is opened in read and write modes simultaneously.

Cache hit rate →	0%	50%	90%	95%	98%	100%	Fuse/Null
Stats per second	5774	7186	8871	9851	11879	23652	36042
Time per stat (μs)	173.2	139.2	112.7	101.5	84.2	42.2	27.7

As can be seen from this table, the procap cache is extremely helpful in attaining efficiency. The difference of the time values in the last two columns is an estimate of the time it takes to check a cached procap (i.e. the time needed to check the conditions in a procap). In this case, this time is $42.2 - 27.7 = 14.5\mu$s. This estimate is rough, and the actual time varies with the complexity of the conditions in the procap. In other experiments, we have found that this time varies from 10 to 100μs. By taking the difference of the time values in the first and last columns, we obtain an estimate of the time required to read a procap, check its signature, parse the procap, and check its conditions. In this experiment, this time is $173.2 - 27.7 = 145.5\mu$s. Additional time may be needed to seek to the procap on disk, which was most likely not counted here, since the procaps used were in a single directory in the underlying file system, hence making the latter's cache very effective. Nonetheless, this suggests that, in general, procap checking is dominated by reading and parsing times. The signatures we use for procaps are message authentication codes, which can be verified in 1 to 2μs each.

File creation and deletion. The table below lists the number of create and delete operations per second that are supported by PCFS and Fuse/Null. These are measured by creating and deleting 10,000 files in a single directory.

Operation	PCFS (op/s)	Fuse/Null (op/s)
Create	1386	4738
Delete	1989	15429

PCFS is approximately 3.5 times slower than FUSE/Null in creating files. This is because in this experiment PCFS also created six default procaps for every file created. As a result, the PCFS numbers measure creation of seven times as many files in three separate directories. Deletion in PCFS in this experiment is nearly 7.7 times slower than that in Fuse/Null. This is because when a file is deleted in PCFS, one procap must be looked up, parsed, and checked, and all procaps related to the file must later be deleted. In this case, each file deletion in PCFS corresponds to seven file deletions on the ext3 file system in three different directories. The effect of the procap cache is negligible during these experiments, since the cache size was kept very small as compared to the number of files.

In summary, assuming a low rate of cache misses, the performance of PCFS on common file operations like read, write, stat, and open is comparable to that of Fuse/Null. On the other hand, less common operations like create and delete are slower because procaps must be managed.

Macrobenchmarks. To understand the performance of PCFS in practice, we also ran two simple macrobenchmarks. The first (called OpenSSL in the table below), untars the OpenSSL source code, compiles it and deletes it. The other (called Fuse in the table

below), performs similar operations for the source of the fuse kernel module five times in sequence. As can be seen, the performance penalty for PCFS as compared to Fuse/Null is approximately 10% for OpenSSL, and 2.5% for Fuse. The difference arises because the OpenSSL benchmark depends more on file creation and deletion as compared to the Fuse benchmark.

Benchmark	PCFS (s)	Fuse/Null (s)	Ext3 (s)
OpenSSL	126	114	94
Fuse × 5	79	77	70

In practice, a file system like PCFS may be used for protecting sensitive files, common operations on which (such as viewing and editing through interactive editors) may be far less file operation-intensive than the macrobenchmarks here. In those cases, the performance of PCFS will be closer to that of Fuse/Null and ext3, than reported in the above table.

7.4 Trusted Code Base and Trust Assumptions

We conclude this chapter with a discussion of the trusted code base (TCB) in the implementation of PCFS, and the trust assumptions on the environment in which PCFS operates. Readers should bear in mind that the main purpose of implementing PCFS is only to show that dynamic access policies can be enforced efficiently with procaps; minimizing the trusted computing base and reducing trust assumptions on the environment are not important objectives of this thesis.

Trusted Code Base. For the PCFS front end, the primary trusted code base is the implementation of the proof verifier, which is approximately 5300 lines of SML code including code for parsing and checking certificates in a custom format. In addition, the front end relies on OpenSSL for cryptographic operations. As in all PCA deployments, the automatic prover for BL is not part of the TCB. For a practical deployment, a public-key infrastructure may be used to manage signing and verification keys, which would also become part of the TCB.

The PCFS back end runs over Fuse, which is a module in the Linux kernel. In the broadest sense, therefore, the TCB for the back end includes the entire Linux kernel. Discounting the Linux kernel and the Fuse module, the TCB for the back end includes code for parsing, caching, and checking procaps, and a small amount of stub code for handling kernel upcalls. Together they constitute approximately 10,000 lines of C++ code. The back end also relies on OpenSSL for verification of signatures on procaps.

Trust Assumptions. The PCFS front end relies on the assumption that honest users and, in particular, policy administrators protect their signing keys. If this were not the case then malicious users could forge certificates and gain access. Along similar lines, the user designated admin must be completely trusted because it may create any access policy. Since access rights in PCFS are tied to actual Linux users, it is also assumed that user accounts are securely protected.

PCFS is implemented as a virtual file system and data is protected on the underlying file system (e.g., ext3) by giving its ownership to the superuser. Consequently, three trust assumptions are that (a) The superuser account is securely protected, (b) The superuser is trustworthy, and (c) The access control mechanisms on the underlying file system work correctly for data owned by the superuser. Further, we must also rely on the environment to ensure that file data is not leaked through interfaces besides the file system interface (e.g., data may be leaked through memory maps of files). It is also assumed that the communication between the kernel and the PCFS server listening to kernel upcalls is secure.

Chapter 8

Case Study: Access Control for Classified Information in the U.S.

This chapter is a case study for the use of our proof-carrying file system PCFS and our authorization logic BL. The subject of the case study are policies that control dissemination of classified information in the hands of intelligence and defense agencies in the U.S. We show that these policies, which are quite extensive as well as dynamic due to their reliance on both system state as well as explicit time, can be represented in BL and enforced with PCFS. The content of this chapter also appeared previously in a technical report that was authored jointly with Frank Pfenning, Denis Serenyi, and Brian Witten [69].

The primary source of policies formalized in this chapter is interviews of intelligence personnel conducted by Brian Witten and Denis Serenyi, and provided to the author in the form of five internal reports as part of a government contract. Some parts of policies are based on Executive Orders of the White House [110, 111] or Director of Central Intelligence Directives (DCIDs) [108, 109]. Due to this mixed source of information, we do not explicitly cite our sources again. None of the information on which this chapter relies is classified. Despite the realistic sources of the policies, the chapter should not be construed as an authoritative reference on policies for controlling access to classified information, or of the actual practices followed for their enforcement in intelligence and defense establishments. The only intention of the chapter is to show that BL is expressive enough to encode a large, representative part of the policies, which can then be enforced directly in PCFS.

The rest of this chapter is organized as follows. §8.1 provides an overview of classified information, including the life cycle of a sensitive file and the high level policy rules for access to sensitive files. §8.2 describes the process of file classification, relevant properties of a classified file, and formal rules for establishing these properties. §8.3 presents rules for giving security clearances to individuals; these clearances are necessary to read classified files. §8.4 explains how properties of a classified file and security clearances of individuals interact to allow access. §8.5 summarizes conclusions and observations from the case study. §8.6 lists all logic predicates used in this chapter, together with their intuitive meanings and the sections in which they are defined.

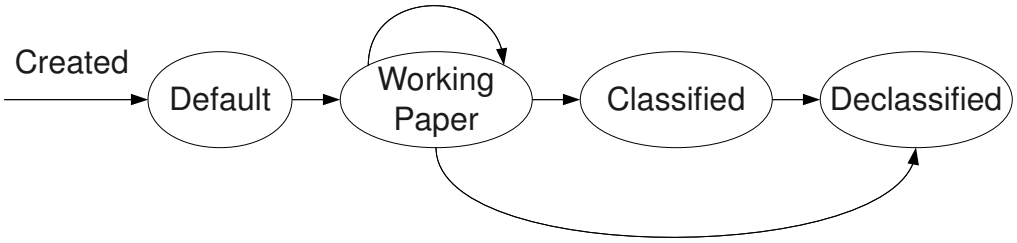

Figure 8.1: States of a sensitive file

Notational conventions. Before proceeding to read this chapter, readers may wish to review the material in §4.1 and §4.3 to refresh their memory about the syntax of BL and conventions for its use in PCFS. We follow a descriptive naming convention for predicates. A predicate name has the form entity/attribute/..., where entity determines the entity whose attribute the predicate describes and attribute is a description of the property the predicate defines. "..." may be any other relevant qualifiers. Common among these qualifiers is h which denotes a helper predicate that is used in the definition of the predicate without the h. As before, interpreted predicates are written in **boldface**.

Recall from §4.3 that policy rules in BL are represented through basic judgments of the form k claims $s \circ [u_1, u_2]$. For brevity, throughout this chapter, we drop the prefix $\circ [u_1, u_2]$ if it is $[-\infty, +\infty]$. Following this convention, most policy rules in this chapter are written as k claims s, when we actually mean k claims $s \circ [-\infty, +\infty]$. Following standard convention from logic programming, variable names starting with uppercase letters are implicitly universally quantified. In general, the universal quantification occurs inside the annotation k claims $\cdot$. So k claims s really means k claims $(\forall \vec{X}.s) \circ [-\infty, +\infty]$, where $\vec{X}$ is the set of all variables starting with uppercase letters in s. Finally, we often write s :- $s_1, \ldots, s_n$ to mean $(s_1 \wedge \ldots \wedge s_n) \supset s$.

8.1 Sensitive Information Life Cycle

This section provides an overview of classification and declassification of information in the U.S. Unfortunately, some of the concepts and methods involved in classification are themselves classified and inaccessible to us. What follows is an abstracted and simplified description of some of the publicly available concepts. The first salient point about classification is that depending on the structure of information, the amount of data classified together may vary: entire files, pages, or paragraphs may be marked for classification as a unit. In this chapter, we assume that the unit of classification is a digital file, since it is at that level that we are interested in controlling access in PCFS.

A typical sensitive file created by an intelligence or defense agency goes through the life cycle depicted in Figure 8.1. There are 4 distinct states, which we discuss below. Transitions between these states are discussed in §8.1.2.

- **Default.** Every newly created file starts in a temporary state, which we call the

default state. Only the individual creating the file has read and write access to a file in this state. A default file may subsequently either be designated a working paper, or it may be deleted.

- **Working paper.** A working paper is a file that will eventually be classified, but whose content has not been finalized. When in this state, read and write access to the file is at the discretion of the agency or group that is working on the file. A file stays in this state for at most 90 days, after which it must either be classified, reviewed again and placed in the same state, declassified, or deleted.

- **Classified.** After the content of a working paper is finalized, it is classified. Read access to a classified file is based on several properties of the classification (e.g., secrecy level, compartments, etc.) that are decided when the file is classified. These properties are discussed in §8.2. In addition, the agency that owns the file must authorize every read access to a classified file. Official guidelines do not specify who, if anyone, has write access to a classified file. Since changing the content of a classified file may require reclassification, it seems reasonable to assume that classified files cannot be written, and we make this assumption throughout this chapter. Owing to concerns of accountability, we also assume that a classified file cannot be deleted.

- **Declassified.** A file may be released to the public (declassified) in two ways: (a) through an executive order, or (b) through an automatic expiration of the classification at a stipulated point of time. In this chapter we make the simplifying assumption that a declassified file may be read by anyone. In actual practice, a file may be declassified to specific groups of people, e.g., U.S. citizens. As for classified files, we assume that declassified files cannot be deleted.

8.1.1 Representation of File State in PCFS

To represent the state of a file, we use extended attributes that are natively supported in PCFS. Recall from §4.3 that the state predicate (**has_xattr** f a v) holds in BL if and only if the file f has the extended attribute named `user.#pcfs.`a set to the term v. v can be any term in BL. We use a specific extended attribute `user.#pcfs.status` with different values to record the state of a file. These values and their meanings are summarized below. The name "`user.#pcfs.status`" is abbreviated to `status` throughout.

Value of extended attribute `status` on file F	Meaning
`default`	F is in default state
`working` T	F is a working paper, put into that state at time T
`classified` T T'	F is classified, effective from time T to time T'
`declassified`	F is declassified

When a file is created PCFS automatically sets its extended attribute status to default, and its owner to the principal who creates the file.[1] The time point T in the value working T represents the time at which the working paper state is effective. This is important because a working paper can be read and written only for 90 days after it enters the state. Similarly, the time point T' in classified T T' is necessary to determine when the classification will expire. If a classification lacks a fixed expiration, the BL constant $+\infty$ may be used for T'.

8.1.2 File State Transition

A central question regarding file states is *who* changes the states of a file. As described in §4.3 and §7.2.1, PCFS supports administrative roles for such purposes by requiring a special permission called govern for modifying any extended attribute of a file whose name starts with the prefix `user.#pcfs.`, or its owner. An individual who has this permission on a file may use either the standard Linux system call `setxattr` or the command line program `setfattr` to change extended attributes of the file. For our proposed enforcement we assume that only a special principal sysadmin (intended to represent a system administrator) is allowed the govern permission on all files, as formalized by the following rule:

admin claims (may sysadmin F govern).

No other rule in our formalization allows the govern permission on any file. Hence sysadmin alone may change the status of a file, and affect its state. Of course, state changes cannot be made ad hoc; sysadmin must perform these transitions only under certain conditions. The conditions necessary for each transition in Figure 8.1 are listed below, together with additional changes that must be made with the transition. Since PCFS only performs access control on files, and policies cannot control the values that file attributes may assume, these conditions cannot be enforced by PCFS. Instead, we must assume that sysadmin follows these guidelines accurately.

- Default to working paper: This transition may be applied at the discretion of the owner (creator) of the file. The status of the file must be set to working T where T is the time at which the transition is applied, and the owner of the file must be changed from the creator of the file to the agency or group that is working on the file (or their representative).

- Working paper to working paper: The purpose of this transition is to extend the 90 day working period of a file. It may be applied at the discretion of the file's owner, which will be the group or agency working on the file. The status of the file must be set to working T where T is the time at which the transition is applied.

- Working paper to declassified: This transition can only be applied after approval from an authority competent to certify that the file does not have information that needs to

[1] As discussed in §7.2.1, in the actual implementation of PCFS, the attribute status is called newfile, and the value default is 1. Since the difference is merely cosmetic, we use the more meaningful names status and default here.

be classified. Such authorities are called Original Classification Authorities or OCAs (see §8.2).

- Working paper to classified: This transition must also be approved by an OCA. In addition, a number of credentials must be issued by different officials to approve this authorization, and to decide the classified file's secrecy level, compartments, etc. (Compartments are discussed in §8.2.2.) These credentials are described in §8.2.4. The status of the file must be set to classified T T', where T is the time of the transition, and T' is determined by the OCA approving the transition.

- Classified to declassified: There is no need to explicitly apply this transition when the classification on a file expires, since the access policies (§8.1.3) automatically allow everyone read access after that time. The transition is needed only to prematurely declassify a file. In that case, approval from an OCA is needed.

8.1.3 Rules for Access to Files

Next we formalize in BL the highest level policy rules for file access. Access to a file depends on its state, and follows the informal guidelines described at the beginning of §8.1. We group our rules by the state to which they apply.

Default. In default state, a file may be read, written, and deleted only by its owner. This is captured by the following rules. The first rule states that it is the admin's policy that if file F is in default state (condition **has_xattr** F status default) and F is owned by K (condition **owner** K F), then K may read F. The second and third rules similarly allow K to write and delete F respectively. The term identity used in the third rule is the PCFS permission needed to delete a file (§7.2.1). We remind the reader that s :- $s_1, \ldots, s_n$ is notation for $(s_1 \wedge \ldots \wedge s_n) \supset s$.

admin claims ((may K F read) :-
has_xattr F status default,
owner F K).

admin claims ((may K F write) :-
has_xattr F status default,
owner F K).

admin claims ((may K F identity) :-
has_xattr F status default,
owner F K).

Working Paper. If a file F is marked as a working paper at time T, then for 90 days after T, F may be read, written, or deleted at the discretion of the owner of file (which, as described in §8.1.2, may be an agency or group). This is enforced by the following rules. $90d$ denotes 90 days, and `is` X E is the special constraint that checks the equality of E and X (see §4.3). The conditions K' says (may K F read), K' says (may K F write), and

K' says (may K F identity) delegate authorization to K', the owner of file F.

admin claims (((may K F read) :-
has_xattr F status (working T),
owner F K',
K' says (may K F read),
is T' $(T + 90d)$) @ $[T, T']$).

admin claims (((may K F write) :-
has_xattr F status (working T),
owner F K',
K' says (may K F write),
is T' $(T + 90d)$) @ $[T, T']$).

admin claims (((may K F identity) :-
has_xattr F status (working T),
owner F K',
K' says (may K F identity),
is T' $(T + 90d)$) @ $[T, T']$).

It is instructive to observe the role of the @ connective in enforcing the 90 day restriction. For example, it is a consequence of the first rule that for any time point $u \in [T, T + 90d]$ at which **has_xattr** F status (working T), **owner** F K', and K' says (may K F read) all hold, admin says (may K F read) also holds. This is not the case if $u \notin [T, T + 90d]$. If 90 days elapse since a file is made a working paper, none of the above rules allow any access to it. In that case, only the principal sysadmin has govern permission to the file (§8.1.2), and this principal must be asked to adjust the status of the file.

Classified. Read access to a classified file is based on properties of its classification such as its secrecy level, compartments, etc., as well as corresponding credentials of the principal to whom access is given. We capture these with the predicate `indi/has-clearances/file` K F which means that principal K's credentials suffice to allow it access to F. A large part of the chapter is devoted to describing how this critical predicate is established; it is defined formally in §8.4. In addition to these properties and credentials, read access to a classified file is contingent on authorization from the file's owner. The following rule specifies this formally.

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

The annotation @ $[T, T']$ restricts the scope of this rule to the duration for which the file is classified. After T', the file is readable by everyone (described next).

Declassified. A file is considered declassified if either its status is marked as such, or if the file is marked classified, but the classification has expired. In both cases, anyone may

read the file. This is captured by the following rules.

admin claims ((may K F read) :-
has_xattr F status declassified).

admin claims (((may K F read) :-
has_xattr F status (classified T T')) @ $[T', +\infty]$).

A consequence of the second rule is that if **has_xattr** F status (classified T T'), then for every time point $u \geq T'$, (admin says (may K F read)) @ $[u, u]$. This does not hold for $u < T'$.

Provisions for Counterintelligence Personnel

In addition to the above rules, there are provisions to allow counterintelligence personnel to read all files that may contain incriminating evidence against an individual they are investigating. Presumably, these provisions apply to files in all states. It is unclear how counterintelligence personnel are assigned to investigate individuals, and how it may be decided whether a file has incriminating evidence against a suspect or not. In our formalization we assume that a special principal oracle can determine these facts accurately. Formally, let the predicate `indi/is-ci` K K' mean that principal K is a counterintelligence officer investigating principal K', and let `indi/is-associated` K' F mean that file F may have incriminating evidence against the suspect principal K'. The following rule states that if the principal oracle states both these predicates, then K may read file F.

admin claims ((may K F read) :-
oracle says (`indi/is-ci` K K'),
oracle says (`indi/is-associated` K' F)).

The principal oracle appears at many places in the rest of this chapter. In each such case, it is assumed to assert relevant facts whose source is either unclear or unspecific from official documents.

8.2 File Classification

In §8.1.2 we stated that when a file is classified, a number of credentials must be issued to determine properties of the file such as its secrecy level, associated compartments, etc. This section explains these credentials in detail, as well as BL rules which combine these credentials to establish properties of a classified file. We start with an intuitive explanation of these properties, and subsequently present BL rules to establish them.

Briefly, there are three relevant properties of a classified file, each of which must be established before the file can be accessed (more precisely, these properties must be known in order to establish the predicate `indi/has-clearances/file` K F from §8.1.3):[2]

[2]It is possible that there are other relevant properties in practice, but these three properties appear to be sufficiently representative.

- Secrecy level: A secrecy level is an indicator of the sensitivity of the contents of a file. It is one of confidential, secret, or topsecret, in increasing order of sensitivity.[3] Read access to a classified file is restricted to individuals who have a secrecy clearance at a level equal to or greater than the secrecy level of the file.

- Citizenship requirement: A set of countries is associated with every classified file. Access is restricted only to citizens of those countries, and to those of the U.S. A commonly used abbreviation is "NOFORN" (no access to foreigners), which corresponds to an empty list of countries.

- Associated compartments: A compartment is a description of the purpose of a file, e.g., a project name or a division within the intelligence community. Every classified file is associated with zero or more compartments. Read access to a classified file is restricted only to those individuals who are associated with at least all compartments that the file is associated with.

Policies for giving clearances to individuals are discussed in §8.3. In this section we discuss rules pertaining to compartment creation and establishment of the properties listed above.

8.2.1 Original Classification Authorities

The authority to decide which file needs to be classified, and what secrecy level, citizenship requirements, and associated compartments a classified file will have rests with very high ranking officers of the executive branch of the government and their representatives. These individuals are called Original Classification Authorities or OCAs. We do not model formally how OCAs are determined. Instead, we assume that the principal oracle (introduced in §8.1.3) names OCAs. Let the predicate `indi/is-oca` O mean that principal O is an OCA. Then the following rule delegates authority over this predicate from `admin` to `oracle`.

admin claims ((`indi/is-oca` O) :-
oracle says (`indi/is-oca` O)).

8.2.2 Compartments

As mentioned earlier, a compartment describes the purpose of information it labels. For example, a compartment may be the name of an intelligence project. Files that have at least one compartment associated with them are called *compartmentalized* files. The purpose of associating a file with compartments is to restrict access to only those individuals who are affiliated with each of those compartments. In addition to restricting access, compartments associated with a classified file play a vital role in determining its secrecy level and citizenship requirements, as we discuss later in this section.

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

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

- A secrecy clearance at level L or higher. Secrecy clearances are described in §8.3.2.
- A background check equivalent to that needed for secrecy clearance at level L' or higher. Background checks are described in §8.3.1.
- A polygraph test if the boolean B is yes. Alternatively, if B is no, then a polygraph test is not necessary to be cleared into C. Polygraph tests are described in §8.3.1.

The following rule delegates the authority to create compartments from admin to every OCA O.

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

SSO and SCG. When a compartment is created, an OCA appoints a special security officer (SSO) to manage the compartment. Afterwards, a set of guidelines for managing all information associated with the compartment is prepared. This set of guidelines is called the compartment's security classification guide (SCG); it must be approved by both an OCA and the SSO of the compartment to which the SCG pertains. Among, other things, the SCG lays down procedures for deciding the secrecy level and citizenship requirements of any file associated with the compartment. As a result, when a file is classified, its associated compartments must be decided first, and subsequently its secrecy level and citizenship requirements must be determined using the SCGs of all the associated compartments.

In our formal model we abstract away the details of an SCG, and treat it only as a symbolic constant. Let the predicate `compartment/has-sso` C S mean that principal S is compartment C's special security officer, and let `compartment/has-scg` C SCG mean that SCG is the security classification guide of compartment C. Then, the first rule below allows an OCA O to assign an SSO S to a compartment C, while the second rule states that both an OCA and the SSO of compartment C must approve C's SCG.

```
admin claims ((compartment/has-sso C S) :-
                indi/is-oca O,
                O says (compartment/has-sso C S)).
```

```
admin claims ((compartment/has-scg C SCG) :-
                                    indi/is-oca O,
                                    O says (compartment/has-scg C SCG),
                                    compartment/has-sso C S,
                                    S says (compartment/has-scg C SCG)).
```

8.2.3 Establishing File Properties

Next, we discuss and formalize rules for determining a file's secrecy level, its citizenship requirements, and its associated compartments. As mentioned in §8.2.2, the compartments associated with a file must be decided first since they are necessary to authorize the file's secrecy level and citizenship requirements.

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

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

The following two rules define the helper predicate `file/has-compartments/h` F CL CL' by induction on CL'. The symbol nil denotes the empty list and | is an infix cons constructor.

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

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

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

Determining a file's secrecy level. As per official guidelines, a file's secrecy level may be set to L if: (a) an OCA says that this should be case, and (b) the SSOs of all compartments associated with the file agree that the SCGs of their respective compartments allow

the file to be given secrecy level L. Formally, let the predicate `file/has-level` F L mean that file F has secrecy level L, and `file/has-level/h` F L CL mean that the SSOs of all compartments in CL agree that F may be given secrecy level L in accordance with their respective SCGs. Then the following rule formally captures the above conditions for giving the secrecy level L to file F.

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

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

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

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

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

The second rule above is an example of exclusive delegation that was introduced in §3.1.2 because the rule gives principal S authority over the predicate `file/has-level/scg`, but the principal admin who gives this authority has no jurisdiction over the predicate.

Determining a file's citizenship requirements. Determining the citizenship requirements for reading a file is similar to determining the file's secrecy level – an OCA must approve the list of countries to whose citizens access should be restricted, and the SSOs of all compartments associated with the file must certify that this list would be allowed by their respective SCGs. Formally, let the predicate `file/has-citizenship` F UL mean that reading file F requires a citizenship of one of the countries in UL (or of the U.S.), `file/has-citizenship/h` F UL CL mean that the SSOs of all compartments in CL agree with this requirement, and `file/has-citizenship/scg` F UL SCG mean that SCG approves this requirement. Then the following three rules may be used to determine a file's citizenship requirements.

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

admin claims (file/has-citizenship/h F UL nil).

admin claims ((file/has-citizenship/h F UL (C' | CL')) :-
compartment/has-sso C' S,
compartment/has-scg C' SCG,
S says (file/has-citizenship/scg F UL SCG),
file/has-citizenship/h F UL CL').

As in the case of rules for determining secrecy levels, if there are no compartments associated with a file F, i.e., if admin says (file/has-compartments F nil), then an OCA O's statement O says (file/has-citizenship F UL) suffices to give a citizenship requirement UL to a file.

8.2.4 Summary of File Classification

As mentioned in §8.1.2, before setting a file F's status attribute to classified T T', the principal sysadmin must ensure that enough credentials are in place to determine the file's secrecy level, citizenship requirements, and associated compartments. The credentials required follow from the rules discussed in §8.2.3, and are summarized below. Although not formalized here, T and T' must also be obtained from an OCA.

- Credentials to determine associated compartments CL
  - An OCA O must issue the credential O claims (file/has-compartments F CL).
  - For every compartment $C \in CL$, the SSO S of C must issue the credential S claims (file/has-compartments F CL).
- Credentials to determine secrecy level L
  - An OCA O must issue the credential O claims (file/has-level F L).
  - For every compartment $C \in CL$, where CL is the list from the previous point, the SSO S of C must issue the credential S claims (file/has-level/scg F L SCG), where SCG is the security classification guide of C.
- Credentials to determine citizenship requirements UL
  - An OCA O must issue the credential O claims (file/has-citizenship F UL).
  - For every compartment $C \in CL$, where CL is the list from the first point, the SSO S of C must issue the credential S claims (file/has-citizenship/scg F UL SCG), where SCG is the security classification guide of C.

In practice, any issued credential will be valid for only a stipulated duration of time. This gets represented in BL through the @ connective. For example, if an OCA O says that file F should have secrecy level L from 2009 to 2011, this would be represented in BL as $(O \text{ claims } (\texttt{file/has-level } F\ L)) \circ [2009{:}01{:}01{:}00{:}00{:}00, 2011{:}12{:}31{:}23{:}59{:}59]$. In general, the validities of all credentials in the list above may be restricted using explicit time. BL's inference rules propagate these time restrictions to other facts derived from the credentials and policy rules.

8.3 Individual Clearances

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

8.3.1 Auxiliary Clearances

Polygraph clearance. Individuals may need to pass a polygraph test to get clearance into certain compartments (§8.2.2 and §8.3.2). Polygraph tests are administered and certified by trained individuals, whom we call polygraph administrators. The procedures for identifying polygraph administrators are beyond the scope of our formalization; we simply assume that oracle names polygraph administrators. Let `indi/is-polygraph-admin` PA mean that principal PA is a trusted polygraph administrator, and let `indi/has-polygraph` K mean that principal K has passed a polygraph test. The following rule states that if oracle says that PA is a polygraph administrator, and PA says that K has passed a polygraph test, then admin will believe the latter.

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

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

From the perspective of formalization, it is very convenient to abstract background checks by the secrecy level for which they are mandatory. Informally speaking, for example,

a *background check at level* confidential would correspond to a background check that is needed to get clearance at secrecy level confidential. This kind of an abstraction is useful because, as per official guidelines, background checks conducted for clearance at secrecy levels expire at fixed intervals of time, and a similar expiration applies to other applications of background checks (e.g., for clearance into compartments). The actual check corresponding to each secrecy level and its expiration time is shown in the table below.

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

Let `indi/is-background-admin` BA mean that principal BA is a background administrator. Further let `indi/has-naclc` K T mean that principal K passed an NACLC at time T,[4] and `indi/has-ssbi` K T mean that principal K passed an SSBI at time T. The following rules define the predicate `indi/has-background` K L, which means that principal K has a background check that is needed for clearance at secrecy level L. There are three rules, one for each possible value of L. A salient point to observe is the use of the @ connective for automatically expiring background checks in accordance with the table above. The symbol y following a number means "years". Hence $15y$ means 15 years. As an example, the first rule below means that if oracle states that BA is a background administrator and BA states that K passed an NACLC at time T, then admin believes that K has a background check at level confidential in the interval $[T, T + 15y]$.

admin claims (((`indi/has-background` K confidential) :-
oracle says (`indi/is-background-admin` BA),
BA says (`indi/has-naclc` K T),
`is` T' $(T + 15y)$) @ $[T, T']$).

admin claims (((`indi/has-background` K secret) :-
oracle says (`indi/is-background-admin` BA),
BA says (`indi/has-naclc` K T),
`is` T' $(T + 10y)$) @ $[T, T']$).

admin claims (((`indi/has-background` K topsecret) :-
oracle says (`indi/is-background-admin` BA),
BA says (`indi/has-ssbi` K T),
`is` T' $(T + 5y)$) @ $[T, T']$).

The remaining policy rules refer only to the predicate `indi/has-background` K L, not to the predicates `indi/has-naclc` K T and `indi/has-ssbi` K T. It is instructive to observe that the @ connective in the above policy rules is placed outside the implication :-. This is important. For example, although perhaps reasonable at a first glance, the following alternate encoding of the third rule is in fact incorrect, as explained below.

[4] In practice, the NACLC for secret clearance may be more extensive than the NACLC for confidential clearance. Even if there is such a distinction, we blur it in our formalization.

admin claims ((indi/has-background K topsecret) @ $[T, T']$:-
oracle says (indi/is-background-admin BA),
BA says (indi/has-ssbi K T),
is T' $(T + 5y)$).

The problem with this alternate rule is that it does not force a relation between the intervals over which oracle says (indi/is-background-admin BA) and BA says (indi/has-ssbi K T) are established and the interval over which admin says (indi/has-background K topsecret) is deduced. In particular, for *any* intervals $[u_1, u_2]$, $[u_1', u_2']$, and $[u_1'', u_2''] \subseteq [T, T + 5y]$, this policy rule together with oracle says (indi/is-background-admin BA) $\circ$ $[u_1, u_2]$ and BA says (indi/has-ssbi K T) $\circ$ $[u_1', u_2']$ implies admin says (indi/has-background K topsecret) $\circ$ $[u_1'', u_2'']$, which is incorrect since, from an informal understanding of the rule, we would expect that $[u_1, u_2] \supseteq [u_1', u_2'] \supseteq [u_1'', u_2'']$ be required.

8.3.2 Primary Clearances

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

Citizenship. We assume that oracle decides the citizenship of each individual. Let indi/has-citizenship K U mean that principal K is a citizen of country U. The following rule delegates authority over this predicate from admin to oracle.

admin claims ((indi/has-citizenship K U) :-
oracle says (indi/has-citizenship K U)).

A useful, related predicate is indi/has-citizenship/list K UL, which means that K is a citizen of at least *one of the countries* in the list UL. The following two rules define this predicate by induction on the list UL.

admin claims ((indi/has-citizenship/list K $(U \mid UL)$) :-
indi/has-citizenship K U).

admin claims
((indi/has-citizenship/list K $(U \mid UL)$) :-
oracle claims (indi/has-citizenship/list K UL)).

Clearance at secrecy levels. As mentioned in §8.3.1, an individual must pass a background check at level L in order to get clearance at secrecy level L. In addition, the individual must have a *need to get the clearance*. Since the factors determining this need are varied and are not completely specified, we simply assume here that oracle may assert this need. Let indi/has-level K L mean that individual K has clearance at secrecy level L, and indi/needs-level K L mean that principal K has a need to get clearance at secrecy level L. level/below L L' means that level L is below the level L' in the order confidential $<$ secret $<$ topsecret. It is defined later. The following rule states that admin will believe that K has clearance at secrecy level L if oracle says that K needs this clearance,

and K has passed a background check at some level L' which is higher than L.

```
admin claims ((indi/has-level K L) :-
                  oracle says (indi/needs-level K L),
                  indi/has-background K L',
                  level/below L L').
```

As formalized in §8.3.1, the validity of `indi/has-background` K L' is limited to 15, 10, or 5 years depending on L'. The above rule and the inference rules of BL transfer the same restrictions to `indi/has-level` K L. The predicate `level/below` is defined by the rules below. Since it is reasonable to assume that all principals agree on the definition of the predicate, these rules are stated by the strongest principal ℓ (Recall that according to BL's inference rules, $(\ell \text{ says } s) \supset (k \text{ says } s)$ for every k and s).

```
ℓ claims (level/below L L).

ℓ claims (level/below confidential secret).

ℓ claims (level/below secret topsecret).

ℓ claims (level/below confidential topsecret).
```

Clearance into compartments. As mentioned in §8.2.2, to be cleared into a compartment, an individual must satisfy all its requirements – secrecy level, background check, and a polygraph test if needed. These requirements are uniquely determined from the predicate `compartment/is` C L L' B, which is established when the compartment C is created. Let the predicates `indi/has-comp-level` K C, `indi/has-comp-background` K C, and `indi/has-comp-polygraph` K C mean that an individual has clearance at an appropriate secrecy level, background check, and polygraph check (if needed) for being cleared into compartment C. The following rules define these predicates by considering respectively the 2nd, 3rd, and 4th arguments of the predicate `compartment/is` C L L' B. An underscore _ represents an implicitly named variable, whose instantiated value is irrelevant to the rule.

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

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

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

Finally, the following two rules define a related, useful predicate `indi/has-compartment/list` K CL which means that K is cleared into *all compartments* in the list CL.

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

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

8.3.3 Summary of Individual Clearances

We close this section with a summary of credentials needed to give various clearances to an individual K.

- Credentials to establish polygraph clearance
  - A polygraph administrator PA must issue the credential PA claims (`indi/has-polygraph` K)
- Credentials to certify background check at level L
  - If L is confidential or secret, then a background administrator BA must issue the credential BA claims (`indi/has-naclc` K T). The check is valid for 15 years after T if $L =$ confidential and for 10 years after T if $L =$ secret.
  - If L is topsecret, then a background administrator BA must issue the credential BA claims (`indi/has-ssbi` K T). The check is valid for 5 years after T.
- Credentials to determine citizenship of country U
  - oracle must issue the credential oracle claims (`indi/has-citizenship` K U).
- Credentials for secrecy clearance at level L

  - oracle must issue the credential oracle claims (`indi/needs-level` K L)
  - Credentials to certify background check at level L or higher as determined by the second point above.

- Credentials for clearance into compartment C established with the predicate `compartment/is` C L L' B
  - oracle must issue the credential oracle claims (`indi/needs-compartment` K C).
  - The SSO S of C must issue the credential S claims (`indi/has-compartment` K C).
  - Credentials for secrecy clearance at level L or higher as determined by the fourth point above.
  - Credentials to certify background check at level L' or higher as determined by the second point above.
  - Credentials to establish polygraph clearance if $B =$ yes as determined by the first point above.

8.4 Clearances to Classified Files

In §8.1.3 we introduced the predicate `indi/has-clearances/file` K F, which means that principal K has enough clearance to read classified file F. Building on other predicates defined in §8.2 and §8.3, we now provide rules that define this critical predicate.

First, we define three auxiliary predicates using the fairly straightforward rules below: (a) `indi/has-level/file` K F, which means that principal K has clearance at a secrecy level higher than that of file F, (b) `indi/has-comps/file` K F, which means that principal K is cleared into all compartments that F is associated with, and (c) `indi/has-cit/file` K F, which means that principal K is a citizen of at least one country in the citizenship requirements of F.

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

The last rule means that any U.S. citizen satisfies the citizenship requirement for reading a file, irrespective of the latter's actual citizenship requirements. The following rule defines

the predicate `indi/has-clearances/file` K F using these three predicates.

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

8.5 Summary

The case study presented in this chapter validates the expressiveness of the logic BL as a framework for expressing authorization policies, and highlights many of its important aspects. The policies presented in this chapter can be enforced directly in the file system PCFS. We conclude this chapter with some salient observations about the case study.

First, the case study exercises a novel feature of BL – interpreted predicates for representing system state. The state of a sensitive file (default, working paper, classified, or declassified) is represented as an extended attribute on the file, which is tested in the logic through the interpreted predicate **has_xattr** F A V in various policy rules that allow access to files (§8.1.3). As discussed in §8.1.2, the PCFS requirement that a special permission govern, distinct from write, be obtained in order to modify extended attributes helps preserve the integrity of file states (see §7.2.1 for a description of PCFS permissions).

Second, the case study relies on BL's support for explicit time not only to model time-bounded certificates, but also to limit the temporal validities of conclusions based on time points present in extended attributes and credentials. Examples of the latter use of explicit time are the 90-day rule for working papers from §8.1.3 and the 5, 10, and 15 year rules for expiration of background checks from §8.3.1. The illustration at the end of §8.3.1 also shows that care must be taken when scoping the @ connective in policies. Seemingly obvious representations of policies with the @ connective may not always be correct.

Besides the dynamic features of BL, namely, interpreted predicates and explicit time, several policy rules presented in this chapter illustrate exclusive delegation (§3.1.2). One example of exclusive delegation was pointed out in §8.2.3; the motif recurs in several other rules as well. As mentioned at the beginning of §3, being able to represent exclusive delegation is the main reason for the use of a new authorization logic BL in this thesis.

Finally, based on the fact that BL is able to express the reasonably complex policies for access to sensitive information, we may expect that policies for information sharing in other organizations (and among them) can also be expressed in BL and enforced in PCFS in a similar manner.

8.6 List of Predicates Used in the Formalization

The following table lists all predicates used in this chapter, the sections of the chapter in which they are described, and their intuitive meanings.

Predicate	Section	Meaning
`compartment/has-scg` C SCG	§8.2.2	SCG is compartment C's security classification guide
`compartment/has-sso` C S	§8.2.2	S is compartment C's special security officer (SSO)
`compartment/is` C L L' B	§8.2.2	C is a compartment, clearance into which requires secrecy clearance at level L, background check at level L', and a polygraph test if $B =$ yes.
`file/has-citizenship` F UL	§8.2.3	Read access to file F is restricted to citizens of countries in the list UL (and of the U.S.)
`file/has-citizenship/h` F UL CL	§8.2.3	The SSOs of all compartments in the list CL certify that read access to F should be restricted to citizens of countries in the list UL (and of the U.S.)
`file/has-citizenship/scg` F UL SCG	§8.2.3	It is conformant with SCG that read access to file F be restricted to citizens of countries in the list UL (and of the U.S.)
`file/has-compartments` F CL	§8.2.3	File F is associated with all compartments in the list CL
`file/has-compartments/h` F CL CL'	§8.2.3	The SSOs of all compartments in the list CL' certify that is okay to associate file F with all compartments in the list CL
`file/has-level` F L	§8.2.3	File F has secrecy level L
`file/has-level/h` F L CL	§8.2.3	The SSOs of all compartments in the list CL certify that is okay to give file F secrecy level L
`file/has-level/scg` F L SCG	§8.2.3	It is conformant with SCG that file F have secrecy level L
has_xattr F A V	§8.1.1	The extended attribute named A on file F is set to value V
`indi/has-background` K L	§8.3.1	Principal K has a background check which is mandatory for clearance at secrecy level L
`indi/has-citizenship` K U	§8.3.2	Principal K is a citizen of country U
`indi/has-citizenship/list` K UL	§8.3.2	Principal K is a citizen of at least one of the countries in the list UL
`indi/has-cit/file` K F	§8.4	Principal K has the citizenship of one of the countries associated with file F (or of the U.S.)
`indi/has-clearances/file` K F	§8.4	Principal K has enough security clearances to read classified file F
`indi/has-compartment` K C	§8.3.2	Principal K is cleared into compartment C
`indi/has-compartment/list` K CL	§8.3.2	Principal K is cleared into all compartments in the list CL

`indi/has-comp-background` K C	§8.3.2	Principal K has passed a background check sufficient for clearance into compartment C
`indi/has-comp-level` K C	§8.3.2	Principal K has secrecy clearance at a level higher than that needed for clearance into compartment C
`indi/has-comp-polygraph` K C	§8.3.2	If clearance into compartment C requires a polygraph test, then principal K has passed one
`indi/has-comps/file` K F	§8.4	Principal K is cleared into all compartments associated with file F
`indi/has-level` K L	§8.3.2	Principal K is cleared at secrecy level L
`indi/has-level/file` K F	§8.4	Principal K has secrecy clearance at a level equal to or above that of file F
`indi/has-naclc` K T	§8.3.1	Principal K passed an NACLC at time T
`indi/has-polygraph` K	§8.3.1	Principal K passed a polygraph test
`indi/has-ssbi` K T	§8.3.1	Principal K passed an SSBI at time T
`indi/is-associated` K F	§8.1.3	File F may potentially have incriminating evidence against principal K
`indi/is-background-admin` BA	§8.3.1	Principal BA is certified to check others' backgrounds
`indi/is-ci` K K'	§8.1.3	Principal K is a counterintelligence officer who is investigating principal K'
`indi/is-oca` O	§8.2.1	Principal O is an Original Classification Authority (OCA)
`indi/is-polygraph-admin` PA	§8.3.1	Principal PA is certified to administer polygraph tests on others
`indi/needs-compartment` K C	§8.3.2	Principal K needs clearance into compartment C
`indi/needs-level` K L	§8.3.2	Principal K needs clearance at secrecy level L
`level/below` L L'	§8.3.2	Secrecy level L is below L' (confidential < secret < topsecret)
`may` K F P	§8.1.3	Principal K has permission P on file F
owner F K	§8.1.3	File F is owned by principal K

Chapter 9

BL^{L}: A Linear Extension of BL

This chapter presents BL^{L}, an extension of BL based on ideas from linear logic [71]. As in linear logic, there are special kinds of hypotheses called *resources* or *linear hypotheses* in hypothetical judgments of BL^{L} that, unlike ordinary hypotheses, do not admit contraction and weakening (Theorem 4.8). Consequently, every resource assumed in a proof must be used in the proof exactly once.[1] Resources are an appropriate way to model *consumable credentials*, i.e. assumptions in an authorization policy that can be used a stipulated number of times only. Such assumptions are useful in practice, e.g., a pay-per-view website may want to give a user a credential that allows her access to a movie only once in return for a fixed amount of money. Using linear hypotheses, which we denote with the letter Λ, consumable credentials can be modeled and enforced through proof-carrying authorization (without procaps) as follows.[2]

1. Certificates establishing consumable hypotheses are distinguishably marked by their creators. In a logical proof consumable credentials are reflected in the linear hypotheses Λ, not the ordinary hypotheses Γ. Each consumable hypothesis is replicated in Λ as many times as it is needed in the proof.

2. By counting the number of occurrences of each consumable credential in the linear hypotheses of each proof it verifies, the proof verifier embedded in the reference monitor tracks the number of times each consumable credential has been used over time. Assuming that the maximum number of times each consumable credential may be used is known to the proof verifier, a proof that relies on more uses of any consumable credential than are still left is immediately rejected.

Linearity plays a crucial role in this enforcement mechanism in two related ways. First, it ensures that consumable credentials are not used in a proof more than the number of

[1] It is important to explain how we count "uses" of a hypothesis. In the so called multiplicative-exponential fragment of linear logic, it is appropriate to say that a hypothesis is used n times in a sequent calculus proof if it appears n times as the principal judgment in either left rules or the rule (init). However, we have to be careful when additive connectives are included, as explained at the end of §9.1.1.

[2] These observations were first made in joint work of the author and others [66] and independently by Cederquist et al. in the setting of auditing traces for access violations [37].

times made explicit in the hypotheses; this is a consequence of a lack of contraction in linear hypotheses. Dually, it ensures that each consumable credential mentioned in the linear hypotheses is actually used in the proof the number of times it is mentioned; this is a consequence of the absence of weakening in linear hypotheses.[3] Together, these two observations imply that the proof verifier is able to accurately track the number of uses of each consumable credential in step (2) above. In the PCFS architecture, this enforcement mechanism has to be modified since the use of consumable credentials in it should be tracked by the back end, not by the proof verifier. Procaps can be used to carry information about consumable credentials used in a proof from the proof verifier to the back end, as described in §9.3.

In addition, linearity can be used to model state, as well as transitions between state, or real expendable resources like money, all of which may be relevant for expressing authorization policies in some cases [54, 66]. The merit of modeling state using linearity (as opposed to interpreted predicates) is that rules for modifying the state can also be expressed and reasoned about within the logic. The disadvantage is that any rule that tries to only read the state must consume the linear hypothesis that represents the state, and then regenerate it. This is awkward in some cases, so in BL^L we allow both linearity as well as interpreted predicates, leaving it up to system designers to use whichever one of the two suits the situation better.

Keeping in mind these uses of linearity, the primary objective of this chapter is to present BL's linear extension BL^L, its proof theory, and some of its metatheoretic properties (§9.1). To prevent repetition of concepts from earlier chapters, we limit our discussion of proof theory to a sequent calculus for BL^L even though a natural deduction system for BL^L can also be constructed. The presentation of the sequent calculus derives from a judgmental presentation of intuitionistic linear logic due to Chang et al. [39], and more directly from prior joint work of the author in the context of authorization [54, 66]. The chapter also presents simple examples of the use of linearity to model consumable credentials (§9.2). These examples are only illustrations to explain the expressiveness added by linearity in the context of authorization; larger examples may be found in prior work on the subject [54, 66]. Finally, the chapter proposes a method of enforcement of consumable credentials in the PCFS architecture that relies on proofs in BL^L as well as procaps (§9.3).

9.1 Syntax, Sequent Calculus, and Metatheory

We present the judgments of BL^L and the relations between them first, and then describe the structure of formulas. Rules of the sequent calculus and its metatheory are postponed to §9.1.1 and §9.1.2, respectively. The proof theory of BL^L (in particular its sequent calculus) relies on four distinct basic judgments, two of which – $s \circ [u_1, u_2]$ and k claims $s \circ [u_1, u_2]$ – were already present in BL (§4.2). We list below all four judgments, together with their intuitive meanings.

[3] Readers familiar with linear logic may be aware that the additive connectives $\top$ and $\mathbf{0}$ may cause weakening to become admissible in certain cases. To avoid such problems we disallow these two connectives in BL^L, as should become clear in §9.1.

- $s \circ [u_1, u_2]$: Formula s holds throughout the interval $[u_1, u_2]$, and this fact may be used any number of times (possibly never).

- $s \star [u_1, u_2]$: Formula s holds throughout the interval $[u_1, u_2]$, and this fact must be used once.

- k claims $s \circ [u_1, u_2]$: Principal k claims throughout the interval $[u_1, u_2]$ that formula s holds, and this fact may be used any number of times (possibly never).

- k claims $s \star [u_1, u_2]$: Principal k claims throughout the interval $[u_1, u_2]$ that formula s holds, and this fact must be used once.

The judgments containing $\star$, which we read as "in", are linear. As their descriptions suggest, they correspond to resources, which if assumed in a proof must be used exactly once in it. (Such judgments may be replicated in the linear hypotheses if they have to be used multiple times.) Hypotheses in BL^{L} are of two types, *unrestricted* Γ which contain assumptions of the forms $s \circ [u_1, u_2]$ and k claims $s \circ [u_1, u_2]$, and *linear* Λ which are comprised of assumptions of the forms $s \star [u_1, u_2]$ and k claims $s \star [u_1, u_2]$. Sequents contain both unrestricted and linear hypotheses, as in the grammar below. The conclusion of a sequent is always of the form $s \star [u_1, u_2]$ because hypothetical judgments with other conclusions can be defined as explained later.

Basic Judgments	J	::=	$s \circ [u_1, u_2] \mid k$ claims $s \circ [u_1, u_2] \mid$
			$s \star [u_1, u_2] \mid k$ claims $s \star [u_1, u_2]$
Unrestricted Hypotheses	Γ	::=	$\cdot \mid \Gamma, s \circ [u_1, u_2] \mid \Gamma, k$ claims $s \circ [u_1, u_2]$
Linear Hypotheses	Λ	::=	$\cdot \mid \Lambda, s \star [u_1, u_2] \mid \Lambda, k$ claims $s \star [u_1, u_2]$
Sequents			$\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$

Basic reasoning principles. The sequent form $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ expresses that the basic judgment $s \star [u_1, u_2]$ follows from the assumptions $\Sigma; \Psi; E; \Gamma; \Lambda$. But what about conclusions of the forms $s \circ [u_1, u_2]$, k claims $s \star [u_1, u_2]$, and k claims $s \circ [u_1, u_2]$? Hypothetical reasoning for these forms of conclusions can be defined, as manifest in the following three principles.[4] $\Gamma|$ and $\Lambda|$ denote the restrictions of the hypotheses Γ and Λ to assumptions of the forms k claims $s \circ [u_1, u_2]$ and k claims $s \star [u_1, u_2]$, respectively.

$$\Gamma| = \{(k \text{ claims } s \circ [u_1, u_2]) \in \Gamma\}$$

$$\Lambda| = \{(k \text{ claims } s \star [u_1, u_2]) \in \Lambda\}$$

Time-unrestricted principle. $s \circ [u_1, u_2]$ follows from the assumptions $\Sigma; \Psi; E; \Gamma; \Lambda$ in view ν if $\Lambda = \cdot$ and $\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} s \star [u_1, u_2]$.

[4] It should be noted that these reasoning principles are not explicit rules in the sequent calculus, but are admissible. Allowing these rules as explicit rules complicates the proof of admissibility of cut, and corresponding rules in the natural deduction system for BL^{L} (which we have not presented here) make it difficult, if not impossible, to characterize canonical proofs.

The principle requires Λ to be empty because $s \circ [u_1, u_2]$ is a conclusion that may be used any number of times, and hence it does not follow directly from $s \star [u_1, u_2]$ if the proof of the latter depends on linear hypotheses.

Claim-linear principle. k claims $s \star [u_1, u_2]$ follows from the assumptions $\Sigma; \Psi; E; \Gamma; \Lambda$ in any view ν if $\Lambda = \Lambda|$ and $\Sigma; \Psi; E; \Gamma|; \Lambda| \xrightarrow{k,u_1,u_2} s \star [u_1, u_2]$.

The principle means that, as for BL, a direct proof of k claims $s \star [u_1, u_2]$ should not depend on any assumptions that do not have the prefix k claims $\cdot$. Further, to track linear hypotheses correctly, we also require that $\Lambda = \Lambda|$.

Claim-unrestricted principle. k claims $s \circ [u_1, u_2]$ follows from the assumptions $\Sigma; \Psi; E; \Gamma; \Lambda$ in any view ν if $\Lambda = \cdot$ and $\Sigma; \Psi; E; \Gamma|; \cdot \xrightarrow{k,u_1,u_2} s \star [u_1, u_2]$.

Again, a direct proof of k claims $s \circ [u_1, u_2]$ should not depend on any assumptions that do not have the prefix k claims $\cdot$. Further, since k claims s is a conclusion that may be used any number of times, it must not depend on linear hypotheses, so Λ must be empty.

In addition to these principles, reasoning in BL^{L} also relies on analogues of the time subsumption and the view principle from §4.2.2, which we do not state explicitly.

Formulas. The syntax of BL^{L} formulas r, s is shown below. We allow most connectives of intuitionistic linear logic, the excluded connectives being $\top$ and $\mathbf{0}$. These two connectives are not included because by writing these connectives in policies weakening may be admitted, which, as explained in the opening of this chapter, is undesirable for modeling consumable credentials. As is standard in linear logic, $r \otimes s$ means that available resources entail r and s simultaneously; $\mathbf{1}$ represents absence of any resources; $r \,\&\, s$ means that available resources suffice to prove either r or s, but not both; $r \oplus s$ means that available resources prove either r or s, but it is not known which; $r \multimap s$ means that resource r can be converted to resource s; and $!s$ means that s holds without restrictions on the number of times it may be used. The new connective, k once s, is the linear form of k says s.

$$\begin{array}{rrcl} \text{Formulas} & r, s & ::= & p \mid i \mid c \mid r \otimes s \mid \mathbf{1} \mid r \,\&\, s \mid r \oplus s \mid r \multimap s \mid\ !\, s \mid \\ & & & \forall x{:}\sigma.s \mid \exists x{:}\sigma.s \mid k \text{ says } s \mid k \text{ once } s \mid s \mathbin{@} [u_1, u_2] \end{array}$$

Judgments internalized as formulas. As in BL, all forms of basic judgments in BL^{L} may be internalized into the syntax of formulas. Formulas that internalize each form of basic judgment are listed in Figure 9.1. Also listed in the figure are other judgments that are equivalent to the basic judgments. We use these equivalences to justify some of the inference rules later. The salient observation here is that of the four basic judgments, only $s \star [u_1, u_2]$ is internalized by a single connective; all others are internalized as the composition of two connectives.

Basic judgment	Internalized as the formula ...	Other equivalent judgment(s)
$s \star [u_1, u_2]$	$s \mathbin{@} [u_1, u_2]$	None
$s \circ [u_1, u_2]$	$(!s) \mathbin{@} [u_1, u_2]$	$(!s) \star [u_1, u_2]$
k claims $s \star [u_1, u_2]$	$(k \text{ once } s) \mathbin{@} [u_1, u_2]$	$(k \text{ once } s) \star [u_1, u_2]$
k claims $s \circ [u_1, u_2]$	$(k \text{ says } s) \mathbin{@} [u_1, u_2]$	$(k \text{ says } s) \star [u_1, u_2]$

Figure 9.1: Basic judgments of BL^{L} and their internalization as formulas

9.1.1 Rules of the Sequent Calculus

Next we describe the rules of the sequent calculus for BL. We start with some basic rules that relate the various judgments, and then proceed to explain the left and right rules for each connective. All rules are summarized in Figures 9.2 and 9.3.

Rules relating basic judgments. The rule (init) below allows a conclusion of $p \star [u_1, u_2]$ from the linear hypothesis $p \star [u_1', u_2']$, if $u_1' \leq u_1$ and $u_2 \leq u_2'$. Observe that no other linear hypotheses must be present, which enforces strict use of the latter. As in the rule's homonym from §4.2.4, the conditions $u_1' \leq u_1$ and $u_2 \leq u_2'$ account for subsumption over time intervals. Theorem 9.6 shows that the generalization of this rule to arbitrary formulas (in place of uninterpreted atoms p) is admissible in BL^{L}.

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

The next rule, (copy), allows an unrestricted hypothesis $s \circ [u_1, u_2]$ to be copied into the linear hypotheses as $s \star [u_1, u_2]$. Since this rule may be applied repeatedly to any unrestricted hypothesis, unrestricted hypotheses may be used any number of times in a proof.

$$\frac{\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2]; \Lambda, s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2]; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}\text{copy}$$

The next two rules, (claims) and (lclaims) are BL^{L} analogues of the BL rule (claims) for unrestricted hypotheses and linear hypotheses respectively. In the case of rule (lclaims), the principal judgment k claims $s \star [u_1, u_2]$ is not retained in the premise, which enforces its strict one-time use.

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

$$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma; \Lambda, s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2'] \\ \nu = k', u_b, u_e \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \qquad \Sigma; \Psi \models k \succeq k'\end{array}}{\Sigma; \Psi; E; \Gamma; \Lambda, k \text{ claims } s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\text{lclaims}$$

Connectives !, k says s, and k once s. Rules for the connectives !, k says s, and k once s follow a similar template. The right rules for these connectives are based on the principles (time-unrestricted), (claim-unrestricted), and (claim-linear), respectively. For instance, the right rule for ! (shown below) states that if $\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} s \star [u_1, u_2]$, then $\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} (!s) \star [u_1, u_2]$. This follows immediately from the principle (time-unrestricted) and the equivalence of the two judgments $(!s) \star [u_1, u_2]$ and $s \circ [u_1, u_2]$ (Figure 9.1). The right rules of k says s and k once s are similarly justified. The left rules for the connectives are straightforward: they replace the principal judgment in the conclusion by an equivalent judgment in the premise.

$$\frac{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} s \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} !s \star [u_1, u_2]}\,!\mathrm{R} \qquad \frac{\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2]; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, !s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\,!\mathrm{L}$$

$$\frac{\Sigma; \Psi; E; \Gamma|; \cdot \xrightarrow{k, u_1, u_2} s \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} k \text{ says } s \star [u_1, u_2]}\,\mathsf{saysR} \qquad \frac{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2]; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, k \text{ says } s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\,\mathsf{saysL}$$

$$\frac{\Sigma; \Psi; E; \Gamma|; \Lambda| \xrightarrow{k, u_1, u_2} s \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda| \xrightarrow{\nu} k \text{ once } s \star [u_1, u_2]}\,\mathsf{onceR} \qquad \frac{\Sigma; \Psi; E; \Gamma; \Lambda, k \text{ claims } s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, k \text{ once } s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\,\mathsf{onceL}$$

It should again be noted that the principal judgments in the left rules are not retained in the premises to enforce linearity correctly. The same pattern repeats in the left rules of all connectives below. Another salient observation is that left rules for connectives apply only to judgments of the form $s \star [u_1, u_2]$. Other judgments in hypotheses can be analyzed in the sequent calculus only by promoting them to judgments of this form through the rules (copy), (claims), and (lclaims).

The connective @. The rules for the @ connective are straightforward and follow the corresponding rules in BL.

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \mathbin{@} [u_1, u_2] \star [u_1', u_2']}\,@\mathrm{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda, s \mathbin{@} [u_1', u_2'] \star [u_1, u_2], s \star [u_1', u_2'] \xrightarrow{\nu} r \star [u_1'', u_2'']}{\Sigma; \Psi; E; \Gamma; \Lambda, s \mathbin{@} [u_1', u_2'] \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1'', u_2'']}\,@\mathrm{L}$$

Constraints and interpreted predicates. The rules for constraints and interpreted predicates in BL^{L} are similar to those in BL. The only important point is that in the right

rules, we require that the linear hypotheses be empty.

$$\frac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} c \star [u_1, u_2]}\text{consR} \qquad \frac{\Sigma; \Psi, c; E; \Gamma; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, c \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\text{consL}$$

$$\frac{\Sigma; E \models i}{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} i \star [u_1, u_2]}\text{interR} \qquad \frac{\Sigma; \Psi; E, i; \Gamma; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, i \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\text{interL}$$

Conjunctions $\otimes$ and $\&$. In linear logic, there are two forms of conjunction – $\otimes$ and $\&$, often called multiplicative conjunction and additive conjunction respectively [71]. The meaning of $s_1 \otimes s_2 \star [u_1, u_2]$ is that the available hypotheses simultaneously entail s_1 and s_2 once each for the entire interval $[u_1, u_2]$. Accordingly, in the rule ($\otimes$R), we split the available linear hypotheses disjointly into Λ_1 and Λ_2, and use them to establish $s_1 \star [u_1, u_2]$ and $s_2 \star [u_1, u_2]$ respectively. Dually, in the premise of the left rule, we introduce both s_1 and s_2 simultaneously in the linear hypotheses.

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda_1 \xrightarrow{\nu} s_1 \star [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma; \Lambda_2 \xrightarrow{\nu} s_2 \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda_1, \Lambda_2 \xrightarrow{\nu} s_1 \otimes s_2 \star [u_1, u_2]}\otimes\text{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \star [u_1, u_2], s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \otimes s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\otimes\text{L}$$

On the other hand, $s_1 \,\&\, s_2 \star [u_1, u_2]$ means that the entire hypotheses may be used to establish s_1 once throughout the interval $[u_1, u_2]$, and also s_2 once throughout the interval $[u_1, u_2]$. Consequently, in the rule ($\&$R), we use the same linear hypotheses to establish both $s_1 \star [u_1, u_2]$ and $s_2 \star [u_1, u_2]$. Dually, there are two left rules; one allows $s_1 \star [u_1, u_2]$ to be assumed given the hypothesis $s_1 \,\&\, s_2 \star [u_1, u_2]$, whereas the other allows $s_2 \star [u_1, u_2]$ to be assumed under the same condition.

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \star [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_2 \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \,\&\, s_2 \star [u_1, u_2]}\&\text{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \,\&\, s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\&\ \mathrm{L}_1$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda, s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \,\&\, s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\&\ \mathrm{L}_2$$

The unit of the multiplicative conjunction, $\mathbf{1}$, denotes absence of resources. Accordingly, it can be established as a conclusion if the linear hypotheses are empty (rule ($\mathbf{1}$R)), and as an assumption it may be removed (rule ($\mathbf{1}$L)).

$$\frac{}{\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} \mathbf{1} \star [u_1, u_2]}\mathbf{1}\text{R} \qquad \frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, \mathbf{1} \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\mathbf{1}\text{L}$$

The unit of additive conjunction, called $\top$ in linear logic, is deliberately excluded from BL^{L}. To understand why we exclude $\top$, let us look at the right rule it would have had, were it to be included.

$$\frac{}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} \top \star [u_1, u_2]} \top\mathrm{R}$$

The problem with this rule is that it consumes arbitrary linear hypotheses Λ. Consequently, using $\top$, it is possible to have sequents that admit weakening. As a simple example, if $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \otimes \top$, then for any Λ', it is also the case that $\Sigma; \Psi; E; \Gamma; \Lambda, \Lambda' \xrightarrow{\nu} s \otimes \top$. Clearly, in this case usage of linear hypotheses is not precisely accounted for by the logic. To avoid this problem we do not include $\top$ in BL^{L}.

Disjunction $\oplus$. The judgment $s_1 \oplus s_2 \star [u_1, u_2]$ means that one of $s_1 \star [u_1, u_2]$ and $s_2 \star [u_1, u_2]$ holds from the available hypotheses, but it may not be known precisely which. $\oplus$ is the analogue of disjunction in intuitionistic linear logic.[5] It has two right rules, which allow $s_1 \oplus s_2 \star [u_1, u_2]$ to be established from $s_1 \star [u_1, u_2]$ and $s_2 \star [u_1, u_2]$ respectively. In the left rule, the same conclusion must be established from either disjunct, since it is not known in general which of the two disjuncts holds.

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \oplus s_2 \star [u_1, u_2]} \oplus\mathrm{R}_1 \qquad \frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_2 \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \oplus s_2 \star [u_1, u_2]} \oplus\mathrm{R}_2$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma; \Lambda, s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, s_1 \oplus s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']} \oplus\mathrm{L}$$

The unit of disjunction, called $\mathbf{0}$ in linear logic, is not included in BL^{L} because like $\top$ it makes accounting of resources imprecise.

Implication $\multimap$. In intuitionistic linear logic (without explicit time), the formula $r \multimap s$ means that the linear hypothesis r entails s. In BL^{L}, the meaning of $\multimap$ must also take into account time intervals, in the same way that implication does in BL, resulting in the following rules.

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \le x_1, x_2 \le u_2; E; \Gamma; \Lambda, s_1 \star [x_1, x_2] \xrightarrow{\nu} s_2 \star [x_1, x_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s_1 \multimap s_2 \star [u_1, u_2]} \multimap\mathrm{R}$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda_1 \xrightarrow{\nu} s_1 \star [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma; \Lambda_2, s_2 \star [u_1', u_2'] \xrightarrow{\nu} r \star [u_1'', u_2''] \qquad \Sigma; \Psi \models u_1 \le u_1' \qquad \Sigma; \Psi \models u_2' \le u_2}{\Sigma; \Psi; E; \Gamma; \Lambda_1, \Lambda_2, s_1 \multimap s_2 \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1'', u_2'']} \multimap\mathrm{L}$$

[5] $\oplus$ is often called additive disjunction, because there is also another kind of disjunction called multiplicative disjunction in classical linear logic. The latter is hard to explain intuitionistically, and is therefore not included here [39].

Quantifiers $\forall$ and $\exists$. The rules for quantifiers in BL^{L} also follow their counterparts from BL.

$$\frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} \forall x{:}\sigma.s \star [u_1, u_2]}\forall\mathrm{R} \qquad \frac{\Sigma; \Psi; E; \Gamma; \Lambda, s[t/x] \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2'] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma; \Lambda, \forall x{:}\sigma.s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\forall\mathrm{L}$$

$$\frac{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s[t/x] \star [u_1, u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} \exists x{:}\sigma.s \star [u_1, u_2]}\exists\mathrm{R} \qquad \frac{\Sigma, x{:}\sigma; \Psi; E; \Gamma; \Lambda, s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}{\Sigma; \Psi; E; \Gamma; \Lambda, \exists x{:}\sigma.s \star [u_1, u_2] \xrightarrow{\nu} r \star [u_1', u_2']}\exists\mathrm{L}$$

Derivable and admissible properties. The rules of the sequent calculus of BL^{L} are summarized in Figures 9.2 and 9.3. In the following we list some properties that explain the interaction of @ with the other connectives. $\vdash s$ means that $\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} s \star [u_1, u_2]$ for every Σ, Ψ, Γ, ν, u_1 and u_2, whereas $\nvdash s$ means that this is not the case. $s_1 \equiv s_2$ denotes $(s_1 \multimap s_2)\ \&\ (s_2 \multimap s_1)$, which is the linear analogue of logical equivalence.

1. $\vdash ((u_1 \leq u_1') \otimes (u_2' \leq u_2)) \multimap ((s\ @\ [u_1, u_2]) \multimap (s\ @\ [u_1', u_2']))$
2. $\vdash ((s_1 \otimes s_2)\ @\ [u_1, u_2]) \equiv ((s_1\ @\ [u_1, u_2]) \otimes (s_2\ @\ [u_1, u_2]))$
3. $\vdash ((s_1\ \&\ s_2)\ @\ [u_1, u_2]) \equiv ((s_1\ @\ [u_1, u_2])\ \&\ (s_2\ @\ [u_1, u_2]))$
4. $\vdash ((s_1 \oplus s_2)\ @\ [u_1, u_2]) \equiv ((s_1\ @\ [u_1, u_2]) \oplus (s_2\ @\ [u_1, u_2]))$
5. $\vdash ((\forall x{:}\sigma.s)\ @\ [u_1, u_2]) \equiv (\forall x{:}\sigma.(s\ @\ [u_1, u_2]))$ $\qquad (x \notin u_1, u_2)$
6. $\vdash ((\exists x{:}\sigma.s)\ @\ [u_1, u_2]) \equiv (\exists x{:}\sigma.(s\ @\ [u_1, u_2]))$ $\qquad (x \notin u_1, u_2)$
7. $\vdash \mathbf{1}\ @\ [u_1, u_2]$
8. $\vdash ((s_1 \multimap s_2)\ @\ [u_1, u_2]) \equiv (\forall x_1{:}\mathsf{time}.\forall x_2{:}\mathsf{time}.\ (((u_1 \leq x_1) \otimes (x_2 \leq u_2) \otimes (s_1\ @\ [x_1, x_2])) \multimap (s_2\ @\ [x_1, x_2])))$
9. $\vdash ((k\ \mathsf{says}\ s)\ @\ [u_1, u_2]) \multimap (k\ \mathsf{says}\ (s\ @\ [u_1, u_2]))$
10. $\nvdash (k\ \mathsf{says}\ (s\ @\ [u_1, u_2])) \multimap ((k\ \mathsf{says}\ s)\ @\ [u_1, u_2])$
11. $\vdash ((s\ @\ [u_1, u_2])\ @\ [u_1', u_2']) \equiv (s\ @\ [u_1, u_2])$

The following are some properties of the connectives !, k says s, and k once s.

1. $\vdash (k\ \mathsf{says}\ s) \multimap (k\ \mathsf{once}\ s)$
2. $\nvdash (k\ \mathsf{once}\ s) \multimap (k\ \mathsf{says}\ s)$
3. $\vdash (k\ \mathsf{says}\ s) \multimap ((k\ \mathsf{once}\ (!s))\ \&\ (!(k\ \mathsf{once}\ s)))$
4. $\nvdash ((k\ \mathsf{once}\ (!s))\ \&\ (!(k\ \mathsf{once}\ s))) \multimap (k\ \mathsf{says}\ s)$

$$\frac{\Sigma;\Psi \models u_1' \leq u_1 \qquad \Sigma;\Psi \models u_2 \leq u_2'}{\Sigma;\Psi;E;\Gamma;p \star [u_1',u_2'] \xrightarrow{\nu} p \star [u_1,u_2]}\text{init} \qquad \frac{\Sigma;\Psi;E;\Gamma,s \circ [u_1,u_2];\Lambda,s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma,s \circ [u_1,u_2];\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}\text{copy}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s \circ [u_1,u_2];\Lambda,s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2'] \\ \nu = k',u_b,u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2 \qquad \Sigma;\Psi \models k \succeq k'\end{array}}{\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s \circ [u_1,u_2];\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}\text{claims}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Gamma;\Lambda,s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2'] \\ \nu = k',u_b,u_e \qquad \Sigma;\Psi \models u_1 \leq u_b \qquad \Sigma;\Psi \models u_e \leq u_2 \qquad \Sigma;\Psi \models k \succeq k'\end{array}}{\Sigma;\Psi;E;\Gamma;\Lambda,k\ \mathsf{claims}\ s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\text{lclaims}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} s \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} !s \star [u_1,u_2]}!\mathrm{R} \qquad \frac{\Sigma;\Psi;E;\Gamma,s \circ [u_1,u_2];\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda,!s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}!\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma|;\cdot \xrightarrow{k,u_1,u_2} s \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} k\ \mathsf{says}\ s \star [u_1,u_2]}\mathsf{says}\mathrm{R} \qquad \frac{\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s \circ [u_1,u_2];\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda,k\ \mathsf{says}\ s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\mathsf{says}\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma|;\Lambda| \xrightarrow{k,u_1,u_2} s \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda| \xrightarrow{\nu} k\ \mathsf{once}\ s \star [u_1,u_2]}\mathsf{once}\mathrm{R} \qquad \frac{\Sigma;\Psi;E;\Gamma;\Lambda,k\ \mathsf{claims}\ s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda,k\ \mathsf{once}\ s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\mathsf{once}\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s\ @\ [u_1,u_2] \star [u_1',u_2']}@\mathrm{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda,s\ @\ [u_1',u_2'] \star [u_1,u_2],s \star [u_1',u_2'] \xrightarrow{\nu} r \star [u_1'',u_2'']}{\Sigma;\Psi;E;\Gamma;\Lambda,s\ @\ [u_1',u_2'] \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1'',u_2'']}@\mathrm{L}$$

$$\frac{\Sigma;\Psi \models c}{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} c \star [u_1,u_2]}\mathrm{cons}\mathrm{R} \qquad \frac{\Sigma;\Psi,c;E;\Gamma;\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda,c \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\mathrm{cons}\mathrm{L}$$

$$\frac{\Sigma;E \models i}{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} i \star [u_1,u_2]}\mathrm{inter}\mathrm{R} \qquad \frac{\Sigma;\Psi;E,i;\Gamma;\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda,i \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\mathrm{inter}\mathrm{L}$$

Figure 9.2: BL$^{\mathrm{L}}$: Sequent calculus, part 1

Counting "uses" of a linear hypothesis. Throughout this chapter, we have used the phrase "use of a linear hypothesis" without an explanation of its precise meaning. Now we explain this phrase. In the absence of rules for & and $\oplus$, it is easy to check by structural induction on proofs that if a resource is repeated n times in the linear hypotheses, then it will appear occur exactly n times as the principal judgment of a left rule or the rule (init) in the proof. Consequently, in the absence of the connectives & and $\oplus$, we may say that a linear hypothesis is used n times in a proof if it appears n times as the principal judgment

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda_1 \xrightarrow{\nu} s_1 \star [u_1,u_2] \qquad \Sigma;\Psi;E;\Gamma;\Lambda_2 \xrightarrow{\nu} s_2 \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda_1,\Lambda_2 \xrightarrow{\nu} s_1 \otimes s_2 \star [u_1,u_2]}\otimes\mathrm{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \star [u_1,u_2], s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \otimes s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\otimes\mathrm{L}$$

$$\frac{}{\Sigma;\Psi;E;\Gamma;\cdot \xrightarrow{\nu} \mathbf{1} \star [u_1,u_2]}\mathbf{1}\mathrm{R} \qquad \frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, \mathbf{1} \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\mathbf{1}\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \star [u_1,u_2] \qquad \Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_2 \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \,\&\, s_2 \star [u_1,u_2]}\&\mathrm{R}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \,\&\, s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\&\,\mathrm{L}_1$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda, s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \,\&\, s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\&\,\mathrm{L}_2$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \oplus s_2 \star [u_1,u_2]}\oplus\,\mathrm{R}_1 \qquad \frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_2 \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \oplus s_2 \star [u_1,u_2]}\oplus\,\mathrm{R}_2$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2'] \qquad \Sigma;\Psi;E;\Gamma;\Lambda, s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, s_1 \oplus s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\oplus\mathrm{L}$$

$$\frac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time};\Psi, u_1 \le x_1, x_2 \le u_2;E;\Gamma;\Lambda, s_1 \star [x_1,x_2] \xrightarrow{\nu} s_2 \star [x_1,x_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s_1 \multimap s_2 \star [u_1,u_2]}\multimap\mathrm{R}$$

$$\frac{\begin{array}{c}\Sigma;\Psi;E;\Gamma;\Lambda_1 \xrightarrow{\nu} s_1 \star [u_1',u_2'] \\ \Sigma;\Psi;E;\Gamma;\Lambda_2, s_2 \star [u_1',u_2'] \xrightarrow{\nu} r \star [u_1'',u_2''] \qquad \Sigma;\Psi \models u_1 \le u_1' \qquad \Sigma;\Psi \models u_2' \le u_2\end{array}}{\Sigma;\Psi;E;\Gamma;\Lambda_1,\Lambda_2, s_1 \multimap s_2 \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1'',u_2'']}\multimap\mathrm{L}$$

$$\frac{\Sigma, x{:}\sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s \star [u_1,u_2]}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} \forall x{:}\sigma.s \star [u_1,u_2]}\forall\mathrm{R} \qquad \frac{\Sigma;\Psi;E;\Gamma;\Lambda, s[t/x] \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2'] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Psi;E;\Gamma;\Lambda, \forall x{:}\sigma.s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\forall\mathrm{L}$$

$$\frac{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} s[t/x] \star [u_1,u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma;\Psi;E;\Gamma;\Lambda \xrightarrow{\nu} \exists x{:}\sigma.s \star [u_1,u_2]}\exists\mathrm{R} \qquad \frac{\Sigma, x{:}\sigma;\Psi;E;\Gamma;\Lambda, s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}{\Sigma;\Psi;E;\Gamma;\Lambda, \exists x{:}\sigma.s \star [u_1,u_2] \xrightarrow{\nu} r \star [u_1',u_2']}\exists\mathrm{L}$$

Figure 9.3: BL^{L}: Sequent calculus, part 2

of a left rule or the rule (init) in the proof. This should also be intuitive because left rules analyze (and therefore consume) hypothesis.

We have to be more careful when the connectives & and $\oplus$ are included. For instance, in the rule (&R), the linear hypotheses are replicated in the two premises, and a naive counting of uses of resources as explained above would suggest that each resource appearing n times in the linear hypotheses in the conclusion has been used $2n$ times in the proof (n times in each premise). A similar problem arises for the rule ($\oplus$L). Therefore, when counting the number of uses of a linear hypothesis above either the rule (&R) or the rule ($\oplus$L) in a derivation, the uses in any one premise should be counted. This can be intuitively justified as follows. From the assumption $s_1 \;\&\; s_2 \star [u_1, u_2]$, we are only allowed to obtain one of the conjuncts (rules (& L_1) and (& L_2)), so only one of the premises of a proof ending in (&R) can be used if the proof were to substitute a hypothesis in another proof. The latter becomes clear upon a careful scrutiny of the proof of admissibility of cut (Theorem 9.5). Similarly, the rule ($\oplus$L) corresponds to a proof by cases, and again, only one of its branches will be used when the principal assumption of the rule is substituted by another proof.

9.1.2 Metatheory of the Sequent Calculus

In this section we prove several interesting metatheoretic properties of the sequent calculus of BL^{L}. These properties are generalizations of properties presented in §4.2.5 for the sequent calculus of BL. We start with weakening and contraction, which are stated in the following theorem. Conspicuously, neither weakening nor contraction holds for the linear hypotheses Λ.

Theorem 9.1 (Weakening and Contraction)**.** *The following hold:*

1. *(Weakening)*
   - *(a)* $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *implies* $\Sigma, x{:}\sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$.
   - *(b)* $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *implies* $\Sigma; \Psi, c; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$.
   - *(c)* $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *implies* $\Sigma; \Psi; E, i; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$.
   - *(d)* $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma, J; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$.
2. *(Contraction)* $\Sigma; \Psi; E; \Gamma, J, J; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma, J; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$.

Further the derivation in the consequent of each statement has a depth no more than that of the antecedent.

Proof. By separate induction on the given derivation for each property. □

Theorem 9.2 (Instantiation)**.** $\Sigma, x{:}\sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ *and* $\Sigma \vdash t : \sigma$ *imply* $\Sigma; \Psi[t/x]; E[t/x]; \Gamma[t/x]; \Lambda[t/x] \xrightarrow{\nu[t/x]} s[t/x] \star [u_1[t/x], u_2[t/x]]$

Proof. By induction on the derivation of $\Sigma, x{:}\sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$. □

Theorem 9.3 (View subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$

2. $\nu = k_0, u_b, u_e$
3. $\Sigma; \Psi \models k_0 \succeq k_0'$, $\Sigma; \Psi \models u_b \leq u_b'$, *and* $\Sigma; \Psi \models u_e' \leq u_e$.
4. $\nu' = k_0', u_b', u_e'$

Then $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu'} s \star [u_1, u_2]$ *by a derivation of smaller or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ and case analysis of its last rule. □

Theorem 9.4 (Time subsumption)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$
3. $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_n, u_m]$.

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ and case analysis of its last rule, as in the proof of Theorem 4.11. The proof appeals to Theorem 9.3 for the cases (saysR) and (onceR), and for the case ($\multimap$R), we appeal to a lemma which states that $\Sigma; \Psi \models c$ and $\Sigma; \Psi, c; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$ imply $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$. The latter follows by a straightforward induction on the derivation of $\Sigma; \Psi, c; E; \Gamma; \Lambda \xrightarrow{\nu} s \star [u_1, u_2]$. □

Theorem 9.5 (Admissibility of cut)**.** *The following four properties hold:*

1. *Suppose that*
   (a) $\Sigma; \Psi; E; \Gamma; \Lambda_1 \xrightarrow{\nu} s \star [u_1, u_2]$ *and*
   (b) $\Sigma; \Psi; E; \Gamma; \Lambda_2, s \star [u_1, u_2] \xrightarrow{\nu} s' \star [u_1', u_2']$

   Then $\Sigma; \Psi; E; \Gamma; \Lambda_1, \Lambda_2 \xrightarrow{\nu} s' \star [u_1', u_2']$.
2. *Suppose that*
   (a) $\Sigma; \Psi; E; \Gamma; \cdot \xrightarrow{\nu} s \star [u_1, u_2]$ *and*
   (b) $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2]; \Lambda \xrightarrow{\nu} s' \star [u_1', u_2']$

   Then $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s' \star [u_1', u_2']$.
3. *Suppose that*
   (a) $\Sigma; \Psi; E; \Gamma|; \Lambda_1| \xrightarrow{k, u_1, u_2} s \star [u_1, u_2]$
   (b) $\Sigma; \Psi; E; \Gamma; \Lambda_2, k \ \mathsf{claims}\ s \star [u_1, u_2] \xrightarrow{\nu} s' \star [u_1', u_2']$

Then $\Sigma; \Psi; E; \Gamma; \Lambda_1|, \Lambda_2 \xrightarrow{\nu} s' \star [u_1', u_2']$.

4. *Suppose that*

   (a) $\Sigma; \Psi; E; \Gamma|; \cdot \xrightarrow{k,u_1,u_2} s \star [u_1, u_2]$

   (b) $\Sigma; \Psi; E; \Gamma, k \mathsf{\ claims\ } s \circ [u_1, u_2]; \Lambda \xrightarrow{\nu} s' \star [u_1', u_2']$

   Then $\Sigma; \Psi; E; \Gamma; \Lambda \xrightarrow{\nu} s' \star [u_1', u_2']$.

Proof. By a simultaneous lexicographic induction, first on the size of the cut formula s, then on the orders (2) > (1), (3) > (1), and (4) > (1), and finally on the depths of the two given derivations, as in prior work [43, 54, 66]. □

Theorem 9.6 (Identity)**.** *If* $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$*, then* $\Sigma; \Psi; E; \Gamma; s \star [u_1, u_2] \xrightarrow{\nu} s \star [u_1', u_2']$.

Proof. By induction on s. □

9.2 Examples of Use

In this section we present two examples of policies that use linearity to encode motifs like consumable credentials, and real expendable resources like money. These examples are illustrative, primarily intended to enumerate the expressiveness of BL^{L}. Other examples on the use of linearity in the context of authorization may be found in prior work [54, 66].

Consumable credentials. The simplest use of linearity is in modeling consumable credentials, which we illustrate through a hypothetical example inspired by the Grey system [20]. Assume that access to doors in certain offices is controlled through proof-carrying authorization, and that the owner of an office is allowed to decide who may enter an office. Assuming that the predicate `mayenter` K' K means that K' may enter the office of K, and that admin has ultimate authority on access, the following policy rule gives each principal K authority to decide who may enter her office. (As in §8, variables in uppercase letters are universally quantified immediately after the annotation claims.)

$$\mathsf{admin\ claims}\ ((K\ \mathsf{once}\ (\texttt{mayenter}\ K'\ K)) \multimap (\texttt{mayenter}\ K'\ K)) \circ [-\infty, +\infty]$$

The policy rule is unrestricted (it can be used any number of times) since it contains the symbol $\circ$, not $\star$. Use of the connective once in the policy rule, as opposed to says, is important because it gives every principal the ability to allow another principal access to her office only once. For example, in conjunction with the previous policy rule, the following *consumable credential* issued by Alice would allow Bob to enter Alice's office at most once in the interval in the week January 01, 2009 – January 07, 2009. After Bob uses this credential once, it will be marked consumed by the reference monitor, so Bob will not be able to enter Alice's office again using this credential.

$$\mathsf{Alice\ claims}\ (\texttt{mayenter Bob Alice}) \star [2009{:}01{:}01, 2009{:}01{:}07]$$

It is easy to check that the above two assumptions entail that admin once (`mayenter Bob Alice`) $\star$ [2009:01:01, 2009:01:07] and that this would not be the case if we replace once by says in the first rule. Also observe that both linearity and time restrict the use of the second credential, but in different ways.

Had Alice wanted to allow Bob unlimited access to her office during the week January 01, 2009 – January 07, 2009, she could have issued the following unrestricted credential in place of the second credential above.

$$\text{Alice claims } (\texttt{mayenter Bob Alice}) \circ [2009{:}01{:}01, 2009{:}01{:}07]$$

Using the first and the third assumptions above, it is possible to derive in BL^{L} the judgment admin says (`mayenter Bob Alice`) $\star$ [2009:01:01, 2009:01:07], which would allow Bob to access Alice's office any number of times during the week.

Modeling actual state. Using constraints and linearity, it is possible to model actual system state and to even keep track of expendable resources like money in the logic. For instance, the following rule formalizes the fact that an individual K possessing $\$N$ initially (predicate `hasmoney` K N) may spend \$10 to buy a movie ticket from the theater (formula theater once (`ticket` K)), leaving the individual with $\$(N-10)$.

$$\begin{aligned}((\texttt{hasmoney}\ K\ N) \otimes (N \geq 10) \otimes (\texttt{is}\ N'\ (N-10))) \\ \multimap ((\texttt{hasmoney}\ K\ N') \otimes (\text{theater once } (\texttt{ticket}\ K))) \circ [-\infty, +\infty]\end{aligned}$$

The above rules expresses a *state transition* that reduces the amount of money K has and generates an authorization (ticket) in return. The ticket may be used by the individual to enter the theater, which we express as follows.

$$\text{theater claims } ((\texttt{ticket}\ K) \multimap (\texttt{mayenter}\ K\ \text{theater})) \circ [-\infty, +\infty]$$

As an example, the above two policy rules and the linear hypothesis (`hasmoney` Alice 30) $\star$ $[-\infty, +\infty]$ together entail the following judgment in BL^{L} which means that Alice could access the theater thrice and be left with no money.

$$\begin{aligned}((&\text{theater once } (\texttt{mayenter}\ K\ \text{theater})) \otimes \\ &(\text{theater once } (\texttt{mayenter}\ K\ \text{theater})) \otimes \\ &(\text{theater once } (\texttt{mayenter}\ K\ \text{theater})) \otimes \\ &(\texttt{hasmoney Alice } 0)) \circ [-\infty, +\infty]\end{aligned}$$

In this example, linearity has been used to model an expendable resource (money), but consumption of the resource is not tracked via a reference monitor. Instead a linear predicate `hasmoney` K N that models actual possession of money is consumed and updated by the first rule. More examples of such use of linearity may be found in prior work [66]. It should also be noted that the predicate `ticket` K produced by the first rule and consumed by the second is a consumable credential in the sense mentioned in the opening of this chapter.

9.3 Enforcement with Procaps

In the opening of this chapter we proposed a simple strategy that may be employed to enforce correct use of consumable credentials using BL^L and proof-carrying authorization (without procaps). In that strategy we suggested that consumable credentials used in a proof be assumed in the linear hypotheses, and that the proof verifier keep track of the number of times each consumable credential has been used by counting the number of its occurrences in the linear hypotheses of each proof it successfully verifies. Now we ask whether this strategy can be adapted to PCFS where proofs are verified ahead of access. There are two possibilities for tracking consumable credentials in PCFS:

- The proof verifier may track consumable credentials, and refuse to issue procaps for proofs that rely on exhausted credentials.
- The file system back end may track consumable credentials, relying on the proof verifier to correctly transfer the list of consumable credentials from proofs it verifies to procaps it generates.

The first method, although clearly advantageous in that it does not increase the procap-checking burden of the back end, is not appropriate because it suffers from two related problems. First, it allows a principal to generate a procap using a proof that relies on consumable credentials and to use the procap for access again and again, possibly circumventing the bound on the number of uses of the consumable credentials used in the proof. Second, it is possible to verify a proof, but never use the procap generated from it, thus wasting consumable credentials used in the proof which may have been helpful for proving other authorizations. Owing to these problems, for PCFS, the second method may be more suitable. Accordingly, consumable credentials can be enforced using BL^L in the PCFS architecture as follows.

- Each certificate establishing a consumable credential is distinctly marked by the creator as being consumable. The number of times such a credential can be used is made available to the reference monitor in the PCFS back end. (Details on this follow below.)
- During proof verification, the proof verifier requires that any hypothesis established by a certificate marked as consumable (previous point) be assumed as a resource, possibly repeated many times. The inference rules of BL^L now ensure that each consumable credential is used in the proof exactly the number of times it is repeated in the linear hypotheses. The proof verifier puts the identity of each consumable credential occurring in the linear hypotheses of a proof as well as the number of times it appears in the linear hypotheses into the procap it generates from the proof.
- Whenever the PCFS back end retrieves and checks a procap, it increases (in an internal database) the number of uses of each consumable credential as mentioned in the procap. If the count of any consumable credential exceeds its maximum stipulated use, then the procap is rejected, else its other conditions are checked as discussed in §5.2.

An important question we have not yet addressed is how the PCFS back end learns the identities and maximum stipulated use of each consumable credential. (A similar question arises if the proof verifier tracks use of consumable credentials in proof-carrying authorization without procaps.) One possibility in this regard, which may be easy to implement in a centralized system like the current implementation of PCFS, is to require that creators of consumable credentials report them to the back end, which could maintain a central database of such reports. A procap relying on any consumable credential not reported to the back end could be immediately rejected. In a fully distributed setting, it may be implausible to assume that all consumable credentials ever issued would be reported to the back end. In that case, the back end may contact issuers of credentials on a need-only basis. More so, the issuers may themselves keep track of use of credentials they issue, and update their records when the back end contacts them. However, the latter approach results in an atomicity problem – either all issuers of consumable credentials mentioned in a procap should update their records and the back end must know this so that it can proceed, or none should and the back end must know this so that it can reject the procap. Bowers et al. discuss, implement, and evaluate the performance of contract signing protocols for solving this atomicity problem in the context of proof-carrying authorization without procaps [34].

9.4 Related Work

Linearity was first introduced in a classical logic by Girard [71]. Intuitionistic versions were later considered by several authors, e.g., [9, 27, 140]. In a judgmental form that we build upon, the logic was first presented by Chang et al. [39]. Going beyond linear logic, Wright [144] described a meta-logic in which the uses of each hypothesis are explicitly counted by annotations. Wright's annotations are more general than replication of hypotheses suggested in this chapter because annotations can be used to encode many different logics of resources including linear logic.

In the context of authorization, there has been limited work on the use of linearity. The area was pioneered in the work of the author and others [66], where linearity was considered for expressing not only consumable credentials but also for expressing elements of state, knowledge of individuals, and authorization-guarded transitions on both. The paper also showed that invariant properties of state may be formally expressed and verified in logical proofs. The latter idea has been developed and refined significantly in recent work by DeYoung and Pfenning [55].

Bowers et al. [34] implemented and tested enforcement of consumable credentials using linear logic to track use of consumable credentials in proofs, and contract signing protocols to track their use in the reference monitor. Independent of other work mentioned above, Cederquist et al. [37] developed a logic for auditing authorization violations on traces that contained a limited form of linearity for recording authorizations that may be used once only. Barth and Mitchell [17] proved, in the context of digital rights management, that all strategies for selecting consumable credentials to authorize a request without knowing future requests suffer from a common problem of non-monotonicity: replacing a credential by a more general credential may cause the algorithm to behave worse in the future. Using a

fragment of linear logic for expressing digital rights they further showed that, under certain conditions on formulas representing the rights, monotonicity can be recovered.

A combination of linearity and explicit time was explored earlier in the author's joint work on η logic [54], later developed in great depth in DeYoung's undergraduate thesis [53]. Although the fundamental nature of the says modality in η logic is different from that in BL, and in particular, the linear version of η logic contains only one says connective not two like BL^{L}, many of the ideas that this chapter builds on owe at least a vague allegiance to that work. The direct inspiration for the methods and work in this chapter, however, is the author's earlier joint work [66].

Chapter 10

Conclusion: Directions for Future Work

In this thesis we have introduced proof theory and metatheory in the context of authorization logic, explained their foundational and practical importance through a new logic BL, developed a logic-based, provably correct mechanism for enforcing dynamic policies in operation-intensive systems, and demonstrated feasibility of the latter by implementing and testing it in a new file system, PCFS. We expect that the architecture for authorization proposed in this thesis will be useful in applications besides PCFS, and hope that it leads to formally grounded, efficient enforcement of access access control policies in operation-intensive interfaces. In this final chapter we list some broad themes for future work that either complement or build upon the work of the thesis.

The overall purpose of access control, of which the work in this thesis is a part, is to provide security for sensitive components of computer systems. Successfully attainment of this goal requires careful consideration of not only authentication and authorization mechanisms, but also usability of the access control framework. Accordingly, complementary to the work in this thesis, it may be useful to develop a front end for authoring authorization policies, converting them to logical form, and checking them for well-formedness. Such a front end may be necessary in some cases for at least two reasons. First, it may be unreasonable to expect that system administrators responsible for creating authorization policies would always understand logical syntax, so policy authoring tools that incorporate common policy creation workflows and automatically translate them to logic may be essential. Prior work on the Grey system has considered these issues [19, 20]. Second, it may be necessary to check policies for well-formedness before using them for enforcement. Such checks may not only ensure that principals have not exceeded their jurisdiction in creating policies, but may also ascertain that policies are well-moded to aid automatic proof search. We alluded to the possibility of mode checks and their importance in §6.4. A policy front end may also compile or residuate policy clauses to aid automatic proof search.

Another possible line of work is to use saturating search à la Datalog to find all consequences of policies and to generate all possible procaps through a centralized inference engine, and hence enforce the policies without the use of explicit proofs. Abduction tech-

niques from logic programming [24, 51] may help find relevant conditions (constraints and interpreted predicates) that would allow access from policies that are dynamic. On the subject of enforcement of policies, an important, open question is that of lower bounds on the complexity of distributed tracking of consumable credentials, as discussed in §9.3.

It also seems plausible to analyze policies against meta-level correctness criteria using proof theoretic methods. Beginnings of such analysis were made in prior joint work of the author and Pfenning [67] under the name of non-interference properties, and also in the work of Abadi [5]. It may be of significant practical use to expand the analysis in these papers, to propose a language for expressing meta-level correctness criteria for authorization policies expressed in logic, and to develop analysis for checking against these criteria automatically.

Finally, even though authorization policies establish legitimacy of single accesses, many relevant security properties of systems are the consequence of interaction between several accesses and ensuing state changes. Developing foundational methods that combine authorization policies expressed in logic with information about workflows or programs that govern system behavior in order to establish relevant invariant properties of state, or even arbitrary safety properties is an interesting topic of further research, at least to the author. Some work on the subject already exists, e.g., [25, 55, 66], but a lot still needs to be done to make the methods useful for analysis of practical systems.

Appendix A

Proofs and Other Details from §3

A.1 Axiomatic Proof System for $\mathbf{BL}_S$

In §3.1, we presented some rules and axioms for the axiomatic system. Here, we list all the rules and axioms, including those listed earlier.

$$\frac{\Sigma \vdash_{\mathcal{H}} s}{\Sigma \vdash_{\mathcal{H}} k \text{ says } s}\mathrm{N} \qquad \frac{\Sigma \vdash_{\mathcal{H}} s \supset s' \qquad \Sigma \vdash_{\mathcal{H}} s}{\Sigma \vdash_{\mathcal{H}} s'}\mathrm{mp} \qquad \frac{\Sigma, x{:}\sigma \vdash_{\mathcal{H}} s \supset s' \qquad x \notin s}{\Sigma \vdash_{\mathcal{H}} s \supset \forall x{:}\sigma.s'}\mathrm{U}$$

$$\frac{\Sigma, x{:}\sigma \vdash_{\mathcal{H}} k \text{ says } (s \supset s') \qquad x \notin s, k}{\Sigma \vdash_{\mathcal{H}} k \text{ says } (s \supset \forall x{:}\sigma.s')}\mathrm{F} \qquad \frac{\Sigma, x{:}\sigma \vdash_{\mathcal{H}} s \supset s' \qquad x \notin s'}{\Sigma \vdash_{\mathcal{H}} (\exists x{:}\sigma.s) \supset s'}\mathrm{E}$$

$$\frac{\Sigma, x{:}\sigma \vdash_{\mathcal{H}} k \text{ says } (s \supset s') \qquad x \notin s', k}{\Sigma \vdash_{\mathcal{H}} k \text{ says } ((\exists x{:}\sigma.s) \supset s')}\mathrm{G}$$

Axioms:

$\Sigma \vdash_{\mathcal{H}} (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2))$	(K)
$\Sigma \vdash_{\mathcal{H}} (k \text{ says } s) \supset k' \text{ says } k \text{ says } s$	(I)
$\Sigma \vdash_{\mathcal{H}} k \text{ says } ((k \text{ says } s) \supset s)$	(C)
$\Sigma \vdash_{\mathcal{H}} (k \text{ says } s) \supset k' \text{ says } s$ if $\Sigma \vdash k \succeq k'$	(S)
$\Sigma \vdash_{\mathcal{H}} s \supset (r \supset s)$	(imp1)
$\Sigma \vdash_{\mathcal{H}} (s \supset s') \supset ((s \supset (s' \supset s'')) \supset (s \supset s''))$	(imp2)
$\Sigma \vdash_{\mathcal{H}} s \supset (s' \supset (s \wedge s'))$	(conj1)
$\Sigma \vdash_{\mathcal{H}} (s \wedge s') \supset s$	(conj2)
$\Sigma \vdash_{\mathcal{H}} (s \wedge s') \supset s'$	(conj3)
$\Sigma \vdash_{\mathcal{H}} s \supset (s \vee s')$	(disj1)
$\Sigma \vdash_{\mathcal{H}} s' \supset (s \vee s')$	(disj2)
$\Sigma \vdash_{\mathcal{H}} (s \supset s'') \supset ((s' \supset s'') \supset ((s \vee s') \supset s''))$	(disj3)
$\Sigma \vdash_{\mathcal{H}} \top$	(true)
$\Sigma \vdash_{\mathcal{H}} \bot \supset s$	(false)
$\Sigma \vdash_{\mathcal{H}} (\forall x{:}\sigma.s) \supset s[t/x]$ if $\Sigma \vdash t : \sigma$	(instU)
$\Sigma \vdash_{\mathcal{H}} s[t/x] \supset \exists x{:}\sigma.s$ if $\Sigma \vdash t : \sigma$	(instE)

Next we introduce a proof tool: a generalized axiomatic system, which allows hypotheses. We write $\Sigma;\Gamma \vdash_{\mathcal{G}} s$ to mean that if $\Sigma \vdash_{\mathcal{H}} s'$ for each s' in Γ, then $\Sigma \vdash_{\mathcal{H}} s$. The rules of the generalized axiomatic system are shown below (axioms from above are unchanged).

$$\frac{}{\Sigma;\Gamma, s \vdash_{\mathcal{G}} s}\text{use} \qquad \frac{\Sigma \vdash_{\mathcal{H}} s \text{ is an axiom}}{\Sigma;\Gamma \vdash_{\mathcal{G}} s}\text{ax}$$

$$\frac{\Sigma;\cdot \vdash_{\mathcal{G}} s}{\Sigma;\Gamma \vdash_{\mathcal{G}} k \text{ says } s}\text{N} \qquad \frac{\Sigma;\Gamma \vdash_{\mathcal{G}} s \supset s' \qquad \Sigma;\Gamma \vdash_{\mathcal{G}} s}{\Sigma;\Gamma \vdash_{\mathcal{G}} s'}\text{mp}$$

$$\frac{\Sigma, x{:}\sigma;\Gamma \vdash_{\mathcal{G}} s \supset s' \qquad x \notin \Gamma, s}{\Sigma;\Gamma \vdash_{\mathcal{G}} s \supset \forall x{:}\sigma.s'}\text{U} \qquad \frac{\Sigma, x{:}\sigma;\cdot \vdash_{\mathcal{G}} k \text{ says } (s \supset s') \qquad x \notin \Gamma, s, k}{\Sigma;\Gamma \vdash_{\mathcal{G}} k \text{ says } (s \supset \forall x{:}\sigma.s')}\text{F}$$

$$\frac{\Sigma, x{:}\sigma;\Gamma \vdash_{\mathcal{G}} s \supset s' \qquad x \notin \Gamma, s'}{\Sigma;\Gamma \vdash_{\mathcal{G}} (\exists x{:}\sigma.s) \supset s'}\text{E} \qquad \frac{\Sigma, x{:}\sigma;\cdot \vdash_{\mathcal{G}} k \text{ says } (s \supset s') \qquad x \notin \Gamma, s', k}{\Sigma;\Gamma \vdash_{\mathcal{G}} k \text{ says } ((\exists x{:}\sigma.s) \supset s')}\text{G}$$

Now we prove some basic properties of the generalized axiomatic system.

Lemma A.1 (Basic properties)**.** *The following hold.*

1. *(Weakening)* $\Sigma;\Gamma \vdash_{\mathcal{G}} s$ *implies* $\Sigma;\Gamma,\Gamma' \vdash_{\mathcal{G}} s$.
2. *(Substitution)* $\Sigma;\Gamma \vdash_{\mathcal{G}} s$ *and* $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s'$ *imply* $\Sigma;\Gamma \vdash_{\mathcal{G}} s'$.
3. *(Deduction)* $\Sigma;\Gamma \vdash_{\mathcal{G}} s \supset s'$ *if and only if* $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s'$.
4. *(Equivalence)* $\Sigma;\cdot \vdash_{\mathcal{G}} s$ *if and only if* $\Sigma \vdash_{\mathcal{H}} s$.

Proof. (1) follows by induction on the given derivation. (2) follows by induction on the second given derivation.

The "only if" direction of (3) follows directly from the rules (use) and (mp): given that $\Sigma;\Gamma \vdash_{\mathcal{G}} s \supset s'$, we get $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s \supset s'$ by (1), and $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s$ from rule (use). Using (mp) on the last two derivations, we get $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s'$ as required.

The "if" direction of (3) follows by an induction on the derivation of $\Sigma;\Gamma, s \vdash_{\mathcal{G}} s'$. This is somewhat tedious, but standard.

Each direction of (4) follows by a simple induction on the given derivation. □

A.2 Proof of Theorem 3.13

In this section we present those details of the proof Theorem 3.13 that were not presented in §3.3. In particular we show here that the natural deduction system for BL_S can be simulated in the axiomatic system and that the axiomatic system can be simulated in the sequent calculus.

Lemma A.2 ($2 \Rightarrow 3$ from Theorem 3.13)**.** $\Sigma;\Gamma \vdash^{k} s$ *implies* $\Sigma \vdash_{\mathcal{H}} k \text{ says } (\overline{\Gamma} \supset s)$.

Proof. By Lemma A.1.4, it suffices to show that $\Sigma; \Gamma \vdash^k s$ implies $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (\overline{\Gamma} \supset s)$. We prove this by induction on the derivation of $\Sigma; \Gamma \vdash^k s$, and show some of the interesting cases below. Basic transformation steps in the axiomatic system such as Currying and un-Currying are often performed implicitly.

Case. $\dfrac{}{\Sigma; \Gamma, s \vdash^k s}\text{hyp}$

To show: $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } ((\overline{\Gamma} \wedge s) \supset s)$

1. $\Sigma; \cdot \vdash_{\mathcal{G}} (\overline{\Gamma} \wedge s) \supset s$ (Axiom (conj3))
2. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } ((\overline{\Gamma} \wedge s) \supset s)$ (Rule (N) on 1)

Case. $\dfrac{\Sigma \vdash k \succeq k_0}{\Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s}\text{claims}$

To show: $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s)$

1. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } ((k \text{ says } s) \supset s)$ (Axiom (C))
2. $\Sigma; \cdot \vdash_{\mathcal{G}} ((k \text{ says } s) \supset s) \supset ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s)$ (Simple theorem in $\mathcal{G}$)
3. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (((k \text{ says } s) \supset s) \supset ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s))$ (Rule (N) on 2)
4. $\Sigma; \cdot \vdash_{\mathcal{G}} (k \text{ says } ((k \text{ says } s) \supset s)) \supset k \text{ says } ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s)$ (Axiom (K) and rule (mp) on 3)
5. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s)$ (Rule (mp) on 4,1)
6. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } ((\overline{\Gamma} \wedge (k \text{ says } s)) \supset s)$ (Axiom (S) and rule (mp) on 5)

Case. $\dfrac{\Sigma, x{:}\sigma; \Gamma \vdash^k s}{\Sigma; \Gamma \vdash^k \forall x{:}\sigma.s}\forall\text{I}$

To show: $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (\overline{\Gamma} \supset \forall x{:}\sigma.s)$

1. $\Sigma, x{:}\sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (\overline{\Gamma} \supset s)$ (i.h. on premise)
2. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (\overline{\Gamma} \supset \forall x{:}\sigma.s)$ (Rule (F) on 1)

Case. $\dfrac{\Sigma; \Gamma| \vdash^k s}{\Sigma; \Gamma \vdash^{k_0} k \text{ says } s}\text{saysI}$

To show: $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset k \text{ says } s)$
Let $\Gamma| = k_1 \text{ claims } s_1, \ldots, k_n \text{ claims } s_n$.

1. $\Sigma; \cdot \vdash_{\mathcal{G}} k \text{ says } (((k_1 \text{ says } s_1) \wedge \ldots \wedge (k_n \text{ says } s_n)) \supset s)$ (i.h. on premise)

2. $\Sigma; \cdot \vdash_{\mathcal{G}} ((k \text{ says } k_1 \text{ says } s_1) \wedge \ldots \wedge (k \text{ says } k_n \text{ says } s_n)) \supset k \text{ says } s$
   (Axiom (K) and rule (mp) on 1)
3. $\Sigma; k \text{ says } k_1 \text{ says } s_1, \ldots, k \text{ says } k_n \text{ says } s_n \vdash_{\mathcal{G}} k \text{ says } s$ (Lemma A.1.3 on 2)
4. $\Sigma; \cdot \vdash_{\mathcal{G}} (k_i \text{ says } s_i) \supset k \text{ says } k_i \text{ says } s_i$ (Axiom (I))
5. $\Sigma; k_i \text{ says } s_i \vdash_{\mathcal{G}} k \text{ says } k_i \text{ says } s_i$ (Lemma A.1.3 on 4)
6. $\Sigma; k_1 \text{ says } s_1, \ldots, k_n \text{ says } s_n \vdash_{\mathcal{G}} k \text{ says } s$ (Lemma A.1.2 on 5,3)
7. $\Sigma; \overline{\Gamma} \vdash_{\mathcal{G}} k \text{ says } s$ (Lemma A.1.1 on 6)
8. $\Sigma; \cdot \vdash_{\mathcal{G}} \overline{\Gamma} \supset k \text{ says } s$ (Lemma A.1.3 on 7)
9. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset k \text{ says } s)$ (Rule (N) on 8)

Case. $\dfrac{\Sigma; \Gamma \vdash^{k_0} k \text{ says } s \qquad \Sigma; \Gamma, k \text{ claims } s \vdash^{k_0} s'}{\Sigma; \Gamma \vdash^{k_0} s'}\text{saysE}$

To show: $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset s')$

1. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset k \text{ says } s)$ (i.h. on 1st premise)
2. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } ((\overline{\Gamma} \wedge k \text{ says } s) \supset s')$ (i.h. on 2nd premise)
3. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset (k \text{ says } s) \supset s')$ (Currying on 2)
4. $\Sigma; \cdot \vdash_{\mathcal{G}} (\overline{\Gamma} \supset k \text{ says } s) \supset ((\overline{\Gamma} \supset (k \text{ says } s) \supset s') \supset (\overline{\Gamma} \supset s'))$ (Axiom (imp2))
5. $\Sigma; \cdot \vdash_{\mathcal{G}} (k_0 \text{ says } (\overline{\Gamma} \supset k \text{ says } s)) \supset ((k_0 \text{ says } (\overline{\Gamma} \supset (k \text{ says } s) \supset s')) \supset k_0 \text{ says } (\overline{\Gamma} \supset s'))$
   (Axiom (K), rule (mp) on 4)
6. $\Sigma; \cdot \vdash_{\mathcal{G}} (k_0 \text{ says } (\overline{\Gamma} \supset (k \text{ says } s) \supset s')) \supset k_0 \text{ says } (\overline{\Gamma} \supset s')$ (Rule (mp) on 5,1)
7. $\Sigma; \cdot \vdash_{\mathcal{G}} k_0 \text{ says } (\overline{\Gamma} \supset s')$ (Rule (mp) on 6,3)

□

Lemma A.3 (3 ⇒ 1 from Theorem 3.13)**.** *$\Sigma \vdash_{\mathcal{H}} s$ implies $\Sigma; \cdot \xrightarrow{k_0} s$ for every k_0.*

Proof. We induct on derivation of $\Sigma \vdash_{\mathcal{H}} s$, case analyzing the last rule in it. Some of the interesting cases are shown here.

Case. $\dfrac{\Sigma \vdash_{\mathcal{H}} s}{\Sigma \vdash_{\mathcal{H}} k \text{ says } s}\text{N}$

To show: $\Sigma; \cdot \xrightarrow{k_0} k \text{ says } s$

1. $\Sigma; \cdot \xrightarrow{k} s$ (i.h. on premise)

2. $\Sigma; \cdot \xrightarrow{k_0} k \text{ says } s$ (Rule (saysR) on 1)

Case. $\dfrac{\Sigma \vdash_{\mathcal{H}} s \supset s' \qquad \Sigma \vdash_{\mathcal{H}} s}{\Sigma \vdash_{\mathcal{H}} s'}\text{mp}$

To show: $\Sigma; \cdot \xrightarrow{k_0} s'$

1. $\Sigma; \cdot \xrightarrow{k_0} s \supset s'$ (i.h. on 1st premise)
2. $\Sigma; \cdot \xrightarrow{k_0} s$ (i.h. on 2nd premise)
3. $\Sigma; s \supset s', s \xrightarrow{k_0} s$ (Theorem 3.11)
4. $\Sigma; s \supset s', s, s' \xrightarrow{k_0} s'$ (Theorem 3.11)
5. $\Sigma; s \supset s', s \xrightarrow{k_0} s'$ (Rule ($\supset$L) on 3,4)
6. $\Sigma; s \xrightarrow{k_0} s'$ (Theorem 3.10 on 5,1)
7. $\Sigma; \cdot \xrightarrow{k_0} s'$ (Theorem 3.10 on 6,2)

Case. $\dfrac{\Sigma, x{:}\sigma \vdash_{\mathcal{H}} k \text{ says } (s \supset s') \qquad x \notin s, k}{\Sigma \vdash_{\mathcal{H}} k \text{ says } (s \supset \forall x{:}\sigma.s')}\text{F}$

To show: $\Sigma; \cdot \xrightarrow{k_0} k \text{ says } (s \supset \forall x{:}\sigma.s')$

1. $\Sigma, x{:}\sigma; \cdot \xrightarrow{k_0} k \text{ says } s \supset s'$ (i.h. on premise)
2. $\Sigma, x{:}\sigma; \cdot \xrightarrow{k_0} s \supset s'$ (Inversion on 1)
3. $\Sigma, x{:}\sigma; s \xrightarrow{k_0} s'$ (Inversion on 2)
4. $\Sigma; s \xrightarrow{k_0} \forall x{:}\sigma.s'$ (Rule ($\forall$R) on 3)
5. $\Sigma; \cdot \xrightarrow{k_0} s \supset \forall x{:}\sigma.s'$ (Rule ($\supset$R) on 4)
6. $\Sigma; \cdot \xrightarrow{k_0} k_0 \text{ says } (s \supset \forall x{:}\sigma.s')$ (Rule (saysR) on 5)

Case. $\Sigma \vdash_{\mathcal{H}} (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2))$ Axiom (K)

To show: $\Sigma; \cdot \xrightarrow{k_0} (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2))$

1. $\Sigma; k \text{ claims } (s_1 \supset s_2), k \text{ claims } s_1, s_1 \supset s_2, s_1 \xrightarrow{k} s_1$ (Theorem 3.11)
2. $\Sigma; k \text{ claims } (s_1 \supset s_2), k \text{ claims } s_1, s_1 \supset s_2, s_1, s_2 \xrightarrow{k} s_2$ (Theorem 3.11)

3. $\Sigma; k \text{ claims } (s_1 \supset s_2), k \text{ claims } s_1, s_1 \supset s_2, s_1 \xrightarrow{k} s_2$ (Rule ($\supset$L) on 1,2)
4. $\Sigma; k \text{ claims } (s_1 \supset s_2), k \text{ claims } s_1 \xrightarrow{k} s_2$ (Rule (claims) on 3)
5. $\Sigma; k \text{ claims } (s_1 \supset s_2), k \text{ claims } s_1, k \text{ says } (s_1 \supset s_2), k \text{ says } s_1 \xrightarrow{k_0} k \text{ says } s_2$ (Rule (saysR) on 4)
6. $\Sigma; k \text{ says } (s_1 \supset s_2), k \text{ says } s_1 \xrightarrow{k_0} k \text{ says } s_2$ (Rule (saysL) on 5)
7. $\Sigma; \cdot \xrightarrow{k_0} (k \text{ says } (s_1 \supset s_2)) \supset ((k \text{ says } s_1) \supset (k \text{ says } s_2))$ (Rule ($\supset$R) on 6)

□

A.3 Proofs from §3.5.1

Theorem A.4 (Soundness; Theorem 3.16)**.** *If* $\Sigma; \Gamma \to \gamma$ *in GP logic, then* $\ulcorner \Sigma; \Gamma \to \gamma \urcorner$ *in* BL_S*.*

Proof. By induction on the given sequent calculus proof of $\Sigma; \Gamma \to \gamma$, and case analysis of the last rule. We show some representative cases here.

Case. $\dfrac{}{\Sigma; \Gamma, p \to p}$init

To show: $\Sigma; \ulcorner \Gamma \urcorner, \ell \text{ claims } p \xrightarrow{\ell} p$. This follows immediately from rule (claims) in Figure 3.3.

Case. $\dfrac{\Sigma; \Gamma \to A}{\Sigma; \Gamma \to k \text{ affirms } A}$affirms

To show: $\Sigma; \ulcorner \Gamma \urcorner \xrightarrow{k} \ell \text{ says } \ulcorner A \urcorner$

1. $\Sigma; \ulcorner \Gamma \urcorner \xrightarrow{\ell} \ulcorner A \urcorner$ (i.h. on premise)
2. $\Sigma; \ulcorner \Gamma \urcorner \xrightarrow{k} \ell \text{ says } \ulcorner A \urcorner$ (Rule (saysR) on 1; $\ulcorner \Gamma \urcorner = \ulcorner \Gamma \urcorner|$)

Case. $\dfrac{\Sigma; \Gamma, A \supset B \to A \qquad \Sigma; \Gamma, A \supset B, B \to C}{\Sigma; \Gamma, A \supset B \to C}\supset$L

To show: $\Sigma; \ulcorner \Gamma \urcorner, \ell \text{ claims } ((\ell \text{ says } \ulcorner A \urcorner) \supset (\ell \text{ says } \ulcorner B \urcorner)) \xrightarrow{\ell} \ulcorner C \urcorner$

1. $\Sigma; \ulcorner \Gamma \urcorner, \ell \text{ claims } ((\ell \text{ says } \ulcorner A \urcorner) \supset (\ell \text{ says } \ulcorner B \urcorner)) \xrightarrow{\ell} \ulcorner A \urcorner$ (i.h. on 1st premise)
2. $\Sigma; \ulcorner \Gamma \urcorner, \ell \text{ claims } ((\ell \text{ says } \ulcorner A \urcorner) \supset (\ell \text{ says } \ulcorner B \urcorner)) \xrightarrow{\ell} \ell \text{ says } \ulcorner A \urcorner$ (Rule (saysR) on 1)
3. $\Sigma; \ulcorner \Gamma \urcorner, \ell \text{ claims } ((\ell \text{ says } \ulcorner A \urcorner) \supset (\ell \text{ says } \ulcorner B \urcorner)), \ell \text{ claims } \ulcorner B \urcorner \xrightarrow{\ell} \ulcorner C \urcorner$

(i.h. on 2nd premise)

4. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), \ell \mathsf{\ says\ } \ulcorner B\urcorner \xrightarrow{\ell} \ulcorner C\urcorner$

(Rule (saysL) on 3)

5. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), (\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner), \ell \mathsf{\ says\ } \ulcorner B\urcorner \xrightarrow{\ell} \ulcorner C\urcorner$ (Theorem 3.8 on 4)

6. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), (\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner) \xrightarrow{\ell} \ulcorner C\urcorner$

(Rule ($\supset$L) on 2,5)

7. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)) \xrightarrow{\ell} \ulcorner C\urcorner$ (Rule (claims) on 6)

Case. $\dfrac{\Sigma; \Gamma, A \supset B \to A \qquad \Sigma; \Gamma, A \supset B, B \to k \mathsf{\ affirms\ } C}{\Sigma; \Gamma, A \supset B \to k \mathsf{\ affirms\ } C} \supset\text{L}$

To show: $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)) \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$

1. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)) \xrightarrow{\ell} \ulcorner A\urcorner$ (i.h. on 1st premise)

2. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)) \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner A\urcorner$ (Rule (saysR) on 1)

3. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), \ell \mathsf{\ claims\ } \ulcorner B\urcorner \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$

(i.h. on 2nd premise)

4. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), \ell \mathsf{\ says\ } \ulcorner B\urcorner \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$

(Rule (saysL) on 3)

5. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), (\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner), \ell \mathsf{\ says\ } \ulcorner B\urcorner \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$ (Theorem 3.8 on 4)

6. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)), (\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner) \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$ (Rule ($\supset$L) on 2,5)

7. $\Sigma; \ulcorner\Gamma\urcorner, \ell \mathsf{\ claims\ } ((\ell \mathsf{\ says\ } \ulcorner A\urcorner) \supset (\ell \mathsf{\ says\ } \ulcorner B\urcorner)) \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner C\urcorner$ (Rule (claims) on 6)

Case. $\dfrac{\Sigma; \Gamma \to k \mathsf{\ affirms\ } A}{\Sigma; \Gamma \to k \mathsf{\ says\ } A} \mathsf{saysR}$

To show: $\Sigma; \ulcorner\Gamma\urcorner \xrightarrow{\ell} k \mathsf{\ says\ } \ell \mathsf{\ says\ } \ulcorner A\urcorner$

1. $\Sigma; \ulcorner\Gamma\urcorner \xrightarrow{k} \ell \mathsf{\ says\ } \ulcorner A\urcorner$ (i.h. on premise)

2. $\Sigma; \ulcorner\Gamma\urcorner \xrightarrow{\ell} k \mathsf{\ says\ } \ell \mathsf{\ says\ } \ulcorner A\urcorner$ (Rule (saysR) on 1; $\ulcorner\Gamma\urcorner = \ulcorner\Gamma\urcorner|$)

□

A.4 Proofs from §3.5.2

The objective of this section is to prove Theorem 3.20 which states that the translation from SL to BL_S described in Figure 3.7 is sound and complete. We start with the proof of soundness.

Lemma A.5 (Soundness)**.** *If $\Delta \vdash_\Gamma g$, then for any fresh constant x, x:*principal*$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} \ulcorner g\urcorner$.*

Proof. By induction on the derivation of $\Delta \vdash_\Gamma g$ and case analysis of the last rule.

Case. $\dfrac{(\forall x_1 \ldots x_n.\ (p \text{ :- } g_1, \ldots, g_m)) \in \Delta \qquad \mathtt{dom}(\theta) \supseteq x_1 \ldots x_n \qquad (\Delta \vdash_\Gamma g_i\theta)_{i\in\{1,\ldots,m\}}}{\Delta \vdash_\Gamma p\theta}\mathrm{bc}$

To show: x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} p\theta$

1. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} xg_i\theta$ (i.h. on 2nd premise)
2. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} (g_1 \wedge \ldots \wedge g_m)\theta$ (Rule ($\wedge$R) on 1)
3. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner, p\theta \xrightarrow{x} p\theta$ (Rule (init))
4. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner, ((g_1 \wedge \ldots \wedge g_m) \supset p)\theta \xrightarrow{x} p\theta$ (Rule ($\supset$L) on 2,3)
5. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner, \forall x_1 \ldots x_n.\ ((g_1 \wedge \ldots \wedge g_m) \supset p) \xrightarrow{x} p\theta$ (Rule ($\forall$L) on 4)
6. x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} p\theta$ (Contraction Theorem 3.8 on 5 using 1st premise)

Case. $\dfrac{(k : \Delta') \in \Gamma \qquad \Delta' \vdash_\Gamma p}{\Delta \vdash_\Gamma k \text{ says } p}\mathrm{says}$

To show: x:principal$;\ulcorner\Gamma\urcorner, \ulcorner\Delta\urcorner \xrightarrow{x} k$ says p
Let $\Delta' = c_1, \ldots, c_n$.

1. y:principal$;\ulcorner\Gamma\urcorner, \ulcorner c_1\urcorner, \ldots, \ulcorner c_n\urcorner \xrightarrow{y} p$ (i.h. on premise for fresh y)
2. $\cdot;\ulcorner\Gamma\urcorner, \ulcorner c_1\urcorner, \ldots, \ulcorner c_n\urcorner \xrightarrow{k} p$ (Theorem 3.7 on 1)
3. $\cdot;\ulcorner\Gamma\urcorner \xrightarrow{k} p$ (Rule (claims) on 2; $(k : \Delta') \in \Gamma$ implies (k claims $\ulcorner c_i\urcorner$) $\in \ulcorner\Gamma\urcorner$)
4. x:principal$;\ulcorner\Gamma\urcorner \xrightarrow{k} p$ (Weakening on 3)
5. x:principal$;\ulcorner\Gamma\urcorner \xrightarrow{x} k$ says p (Rule (saysR) on 4; $\ulcorner\Gamma\urcorner = (\ulcorner\Gamma\urcorner)|)$

□

To prove completeness of the translation, we identify a class of sequents in BL_S, called *regular sequents*, which satisfies two properties: (a) All sequents in the image of the translation $\ulcorner\cdot\urcorner$ are regular, and (b) If the conclusion of any rule from Figure 3.3 is regular, the premises must also be. Then we define an inverse translation $|\cdot|$ from regular sequents to sequents of SL, such that $|\ulcorner\cdot\urcorner|$ is the identity and show that any if a regular sequent is provable in BL_S, then its image under $|\cdot|$ is provable in SL. Completeness of $\ulcorner\cdot\urcorner$ follows immediately.

Before defining regular sequents, we prove basic properties about derivations in SL.

Lemma A.6 (Basic properties of SL)**.** *The following hold in SL*

1. *(Weakening) If* $\Delta \vdash_\Gamma g$ *then* $\Delta, c \vdash_\Gamma g$.
2. *(Contraction) If* $\Delta, c, c \vdash_\Gamma g$ *then* $\Delta, c \vdash_\Gamma g$.
3. *(Substitution) Let* p *be a ground atomic formula. Suppose* $\Delta \vdash_\Gamma p$ *and* $\Delta, p \vdash_\Gamma g$. *Then* $\Delta \vdash_\Gamma g$.

Proof. (1) and (2) follow by straightforward inductions on given derivations. We prove (3) by induction on the derivation of $\Delta, p \vdash_\Gamma g$, and case analysis of the last rule.

Case.

$$\frac{(\forall x_1 \ldots x_n.\ (p' \text{ :- } g_1, \ldots, g_m)) \in \Delta, p \qquad \mathtt{dom}(\theta) \supseteq x_1 \ldots x_n \qquad (\Delta, p \vdash_\Gamma g_i\theta)_{i\in\{1,\ldots,m\}}}{\Delta, p \vdash_\Gamma p'\theta}\,\mathrm{bc}$$

Subcase. $p = \forall x_1 \ldots x_n.\ (p' \text{ :- } g_1, \ldots, g_m)$. Then $n = m = 0$, and $p'\theta = p' = p$. To show: $\Delta \vdash_\Gamma p$. This is already given as an assumption in the statement of the theorem.

Subcase. $p \neq \forall x_1 \ldots x_n.\ (p' \text{ :- } g_1, \ldots, g_m)$. Hence $(\forall x_1 \ldots x_n.\ (p' \text{ :- } g_1, \ldots, g_m)) \in \Delta$. To show: $\Delta \vdash_\Gamma p'\theta$.

1. $(\Delta, p \vdash_\Gamma g_i\theta)_{i\in\{1,\ldots,m\}}$ (i.h. on 3rd premise, m times)
2. $\Delta \vdash_\Gamma p'\theta$ (Rule (bc) on 1 and $(\forall x_1 \ldots x_n.\ (p' \text{ :- } g_1, \ldots, g_m)) \in \Delta$)

□

We also need an inversion lemma about derivations in BL_S.

Lemma A.7 (Strong inversion for ($\wedge$R))**.** *If* $\Sigma; \Gamma \xrightarrow{k} s_1 \wedge s_2$ *in* BL_S, *then for* $i = 1, 2$, $\Sigma; \Gamma \xrightarrow{k} s_i$ *by a shorter or equal derivation.*

Proof. By induction on the given derivation of $\Sigma; \Gamma \xrightarrow{k} s_1 \wedge s_2$. □

Definition A.8 (Regular sequents)**.** We call a BL_S sequent regular if it has the form $\cdot; \Xi, \Theta \xrightarrow{k} \gamma$, where

1. $\gamma = p$ or $\gamma = k'$ says p
2. Ξ contains only hypotheses of the form k_i claims $\ulcorner c_i \urcorner$
3. Θ contains only hypotheses of one of the following two forms:
   (a) Atomic formulas p
   (b) $\forall \vec{x}.\ ((\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p)$, such that for some $\vec{y}$ and substitution θ, (k claims $\forall \vec{y}, \vec{x}.\ ((\ulcorner g'_1 \urcorner \wedge \ldots \wedge \ulcorner g'_m \urcorner) \supset p')) \in \Xi$, $g'_i\theta = g_i$ for each i, and $p'\theta = p$.

Definition A.9 (Inverse translation)**.** The inverse translation $|\cdot|$ from regular sequents and their components to SL, and a subsidiary translation $|\cdot|_k$ on hypotheses are defined as follows.

$$\begin{array}{rcl} |p| & = & p \\ |k \text{ says } p| & = & k \text{ says } p \\ |\Xi| & = & \bigcup_k k : \{c \mid k \text{ claims } \ulcorner c \urcorner \in \Xi\} \\ |\Xi|_k & = & \bigcup\{c \mid k \text{ claims } \ulcorner c \urcorner \in \Xi\} \\ |\Theta| & = & \{p \mid p \in \Theta\} \\ |\cdot; \Xi, \Theta \xrightarrow{k} \gamma| & = & |\Xi|_k, |\Theta| \vdash_{|\Xi|} |\gamma| \end{array}$$

Lemma A.10 (Simulation)**.** *Let* $\cdot; \Xi, \Theta \xrightarrow{k} \gamma$ *be regular and provable in* BL_S*. Then* $|\Xi|_k, |\Theta| \vdash_{|\Xi|} |\gamma|$ *is provable in SL.*

Proof. By induction on the depth of the given sequent calculus proof of $\cdot; \Xi, \Theta \xrightarrow{k} \gamma$ and case analysis of its last rule. Some representative cases are shown here. We often use parantheses in the hypotheses to separate assumptions in Ξ from those in Θ.

Case. $\dfrac{}{\cdot; \Xi, (\Theta, p) \xrightarrow{k} p}$init

To show: $|\Xi|_k, |\Theta|, p \vdash_{|\Xi|} p$.
This follows immediately by rule (bc) on principal formula p.

Case. $\dfrac{\cdot \vdash k \succeq k_0 \qquad \cdot; (\Xi, k \text{ claims } \ulcorner c \urcorner), (\Theta, \ulcorner c \urcorner) \xrightarrow{k_0} \gamma}{\cdot; (\Xi, k \text{ claims } \ulcorner c \urcorner), \Theta \xrightarrow{k_0} \gamma}$claims

By our assumption on principals, $k = k_0$. To show: $|\Xi|_k, c, |\Theta| \vdash_{|\Xi|} |\gamma|$

Subcase. $c = p$.

1. $|\Xi|_k, c, |\Theta|, c \vdash_{|\Xi|} |\gamma|$ (i.h. on premise)
2. $|\Xi|_k, c, |\Theta| \vdash_{|\Xi|} |\gamma|$ (Contraction Lemma A.6 on 1)

Subcase. $c \neq p$

1. $|\Xi|_k, c, |\Theta| \vdash_{|\Xi|} |\gamma|$ (i.h. on premise)

Case. $\dfrac{\cdot; \Xi|, \Theta| \xrightarrow{k} p}{\cdot; \Xi, \Theta \xrightarrow{k_0} k \text{ says } p}\text{saysR}$

To show: $|\Xi|_{k_0}, |\Theta| \vdash_{|\Xi|} k \text{ says } p$

1. $\Xi| = \Xi$ and $\Theta| = \cdot$ (Defn.)
2. $\cdot; \Xi \xrightarrow{k} p$ (premise and 1)
3. $|\Xi|_k \vdash_{|\Xi|} p$ (i.h. on 2)
4. $k : |\Xi|_k \in |\Xi|$ (Defn.)
5. $|\Xi|_{k_0}, |\Theta| \vdash_{|\Xi|} k \text{ says } p$ (Rule (says) on 4,3)

Case. $\dfrac{\cdot; \Xi, \Theta, (\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p \xrightarrow{k} (\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \quad \cdot; \Xi, \Theta, (\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p, p \xrightarrow{k} \gamma}{\cdot; \Xi, \Theta, (\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p \xrightarrow{k} \gamma}\supset\text{L}$

Subcase. $m = 0$. To show: $|\Xi|_k, |\Theta|, p \vdash_{|\Xi|} |\gamma|$

1. $|\Xi|_k, |\Theta|, p, p \vdash_{|\Xi|} |\gamma|$ (i.h. on 2nd premise)
2. $|\Xi|_k, |\Theta|, p \vdash_{|\Xi|} |\gamma|$ (Contraction Lemma A.6 on 1)

Subcase. $m \neq 0$. To show: $|\Xi|_k, |\Theta| \vdash_{|\Xi|} |\gamma|$.
By regularity, there must be some $\vec{y}$ and θ such that k claims $(\forall \vec{y}.\ ((\ulcorner g'_1 \urcorner \wedge \ldots \wedge \ulcorner g'_m \urcorner) \supset p')) \in \Xi$ and $g'_i\theta = g_i$ and $p'\theta = p$. This also implies that $(\forall \vec{y}.\ (p' \text{ :- } g'_1, \ldots, g'_m)) \in |\Xi|_k$.

1. $\cdot; \Xi, \Theta, (\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p \xrightarrow{k} \ulcorner g_i \urcorner$ (Lemma A.7 on 1st premise)
2. $\cdot; |\Xi|_k, |\Theta| \vdash_{|\Xi|} g_i$ (i.h. on 1)
3. $\cdot; |\Xi|_k, |\Theta| \vdash_{|\Xi|} g'_i\theta$ $(g'_i\theta = g_i)$
4. $\cdot; |\Xi|_k, |\Theta| \vdash_{|\Xi|} p'\theta$ (Rule (bc) on 3; $(\forall \vec{y}.\ (p' \text{ :- } g'_1, \ldots, g'_m)) \in |\Xi|_k$)
5. $\cdot; |\Xi|_k, |\Theta| \vdash_{|\Xi|} p$ $(p'\theta = p)$
6. $\cdot; |\Xi|_k, |\Theta|, p \vdash_{|\Xi|} |\gamma|$ (i.h. on 2nd premise)
7. $|\Xi|_k, |\Theta| \vdash_{|\Xi|} |\gamma|$ (Substitution Lemma A.6 on 5,6)

Case. $\dfrac{\cdot; \Xi, \Theta, \forall x, \vec{x'}.((\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p), (\forall \vec{x'}.((\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p))[t/x], \xrightarrow{k} \gamma}{\cdot; \Xi, \Theta, \forall x, \vec{x'}.((\ulcorner g_1 \urcorner \wedge \ldots \wedge \ulcorner g_m \urcorner) \supset p) \xrightarrow{k} \gamma}\forall\text{L}$

To show: $|\Xi|_k, |\Theta| \vdash_{|\Xi|} |\gamma|$

First observe that the premise is regular. Then, the required statement follows by i.h. on the premise. □

$$\frac{}{\Sigma;\Gamma,p\to p}\text{init} \qquad \frac{\Sigma;\Gamma\to s \qquad \Sigma;\Gamma\to s'}{\Sigma;\Gamma\to s\wedge s'}\wedge\text{R} \qquad \frac{\Sigma;\Gamma,s\wedge s',s,s'\to r}{\Sigma;\Gamma,s\wedge s'\to r}\wedge\text{L}$$

$$\frac{\Sigma;\Gamma\to s}{\Sigma;\Gamma\to s\vee s'}\vee\text{R}_1 \qquad \frac{\Sigma;\Gamma\to s'}{\Sigma;\Gamma\to s\vee s'}\vee\text{R}_2 \qquad \frac{\Sigma;\Gamma,s\vee s',s\to r \qquad \Sigma;\Gamma,s\vee s',s'\to r}{\Sigma;\Gamma,s\vee s'\to r}\vee\text{L}$$

$$\frac{}{\Sigma;\Gamma\to\top}\top\text{R} \qquad \frac{}{\Sigma;\Gamma,\bot\to r}\bot\text{L}$$

$$\frac{\Sigma;\Gamma,s\to s'}{\Sigma;\Gamma\to s\supset s'}\supset\text{R} \qquad \frac{\Sigma;\Gamma,s\supset s'\to s \qquad \Sigma;\Gamma,s\supset s',s'\to r}{\Sigma;\Gamma,s\supset s'\to r}\supset\text{L}$$

$$\frac{\Sigma,x{:}\sigma;\Gamma\to s}{\Sigma;\Gamma\to\forall x{:}\sigma.s}\forall\text{R} \qquad \frac{\Sigma;\Gamma,\forall x{:}\sigma.s,s[t/x]\to r \qquad \Sigma\vdash t:\sigma}{\Sigma;\Gamma,\forall x{:}\sigma.s\to r}\forall\text{L}$$

$$\frac{\Sigma;\Gamma\to s[t/x] \qquad \Sigma\vdash t:\sigma}{\Sigma;\Gamma\to\exists x{:}\sigma.s}\exists\text{R} \qquad \frac{\Sigma,x{:}\sigma;\Gamma,\exists x{:}\sigma.s,s\to r}{\Sigma;\Gamma,\exists x{:}\sigma.s\to r}\exists\text{L}$$

Figure A.1: Cut-free sequent calculus for intuitionistic first-order logic

Theorem A.11 (Correctness; Theorem 3.20)**.** *Suppose $k:\Delta\in\Gamma$. Then, $\Delta\vdash_\Gamma g$ in SL if and only if $\cdot;\ulcorner\Gamma\urcorner,\ulcorner\Delta\urcorner\xrightarrow{k}\ulcorner g\urcorner$ in BL_S.*

Proof. Suppose $\Delta\vdash_\Gamma g$. By Lemma A.5, we get for any fresh variable x that $x{:}\mathsf{principal};\ulcorner\Gamma\urcorner,\ulcorner\Delta\urcorner\xrightarrow{x}\ulcorner g\urcorner$. By Theorem 3.7, $\cdot;\ulcorner\Gamma\urcorner,\ulcorner\Delta\urcorner\xrightarrow{k}\ulcorner g\urcorner$.

Conversely, suppose $\cdot;\ulcorner\Gamma\urcorner,\ulcorner\Delta\urcorner\xrightarrow{k}\ulcorner g\urcorner$. Since $k:\Delta\in\Gamma$, this is a regular sequent. Hence by Lemma A.10, we must have $|\ulcorner\Gamma\urcorner|_k,|\ulcorner\Delta\urcorner|\vdash_{|\ulcorner\Gamma\urcorner|}|\ulcorner g\urcorner|$. Next observe that because $k:\Delta\in\Gamma$, $|\ulcorner\Gamma\urcorner|_k=\Delta$. Further by definition, $\Delta\supseteq|\ulcorner\Delta\urcorner|$. Therefore, using contraction (Theorem 3.8), we get $\Delta\vdash_{|\ulcorner\Gamma\urcorner|}|\ulcorner g\urcorner|$. Finally, $|\ulcorner\Gamma\urcorner|=\Gamma$ and $|\ulcorner g\urcorner|=g$. Therefore, $\Delta\vdash_\Gamma g$. □

A.5 Proofs from §3.6

In this section we prove Theorem 3.21, which states that the translation $[\![\cdot]\!]$ from the Horn fragment of BL_S (Figure 3.8) to first-order logic is sound and complete. We use the fairly standard sequent calculus for first-order logic shown in Figure A.1. This sequent calculus admits the usual structural properties of weakening and contraction as well as the cut principle.

If $p=P\ t_1\dots t_n$ is an atomic formula, we write $p\ k$ as an abbreviation for $P\ k\ t_1\dots t_n$. We start by proving an important lemma about the translation, which is needed in the proof of soundness.

Lemma A.12 (Soundness)**.** *If a sequent* $\Sigma; \Delta, \Xi \xrightarrow{k} g$ *in the Horn fragment is provable in* BL_S *then its translation* $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$ *in first-order logic.*

Proof. By induction on the given derivation of $\Sigma; \Delta \xrightarrow{k} g$ and case analysis of the last rule in it. Some representative cases are shown here. Note that many of the cases do not apply at all due to syntactic restrictions on g and Ξ.

Case. $\dfrac{}{\Sigma; \Delta, \Xi, p \xrightarrow{k} p}\text{init}$

To show: $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k, p\ k \to p\ k$
This follows immediately from rule (init) in first-order logic.

Case. $\dfrac{\Sigma \vdash k \succeq k_0 \qquad \Sigma; \Delta, k\ \mathsf{claims}\ d, \Xi, d \xrightarrow{k_0} g}{\Sigma; \Gamma, k\ \mathsf{claims}\ d, \Xi \xrightarrow{k_0} g}\text{claims}$

By assumption on $\succeq$, we must have $k = k_0$. To show: $\Sigma; [\![\Delta]\!], [\![d]\!]_k, [\![\Xi]\!]_k \to [\![g]\!]_k$

1. $\Sigma; [\![\Delta]\!], [\![d]\!]_k, [\![\Xi]\!]_k, [\![d]\!]_k \to [\![g]\!]_k$ (i.h. on premise)
2. $\Sigma; [\![\Delta]\!], [\![d]\!]_k, [\![\Xi]\!]_k \to [\![g]\!]_k$ (Contraction on 1)

Case. $\dfrac{\Sigma; \Delta, \Xi \xrightarrow{k} g \qquad \Sigma; \Delta, \Xi \xrightarrow{k} g'}{\Sigma; \Delta, \Xi \xrightarrow{k} g \wedge g'}\wedge\text{R}$

To show: $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k \wedge [\![g']\!]_k$

1. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$ (i.h. on 1st premise)
2. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g']\!]_k$ (i.h. on 2nd premise)
3. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k \wedge [\![g']\!]_k$ (Rule ($\wedge$R) on 1,2)

Case. $\dfrac{\Sigma; \Delta, \Xi, g \supset d \xrightarrow{k} g \qquad \Sigma; \Delta, \Xi, g \supset d, d \xrightarrow{k} g'}{\Sigma; \Delta, \Xi, g \supset d \xrightarrow{k} g'}\supset\text{L}$

To show: $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k, [\![g]\!]_k \supset [\![d]\!]_k \to [\![g']\!]_k$

1. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k, [\![g]\!]_k \supset [\![d]\!]_k \to [\![g]\!]_k$ (i.h. on 1st premise)
2. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k, [\![g]\!]_k \supset [\![d]\!]_k, [\![d]\!]_k \to [\![g']\!]_k$ (i.h. on 2nd premise)
3. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k, [\![g]\!]_k \supset [\![d]\!]_k \to [\![g']\!]_k$ (Rule ($\supset$L) on 1,2)

Case. $\dfrac{\Sigma; \Delta|, \Xi| \xrightarrow{k} g}{\Sigma; \Delta, \Xi \xrightarrow{k_0} k\ \mathsf{says}\ g}\mathsf{says}\text{R}$

To show: $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_{k_0} \to [\![g]\!]_k$

1. $\Sigma; \Delta \xrightarrow{k} g$ (premise, $\Delta| = \Delta$, $\Xi| = \cdot$)
2. $\Sigma; [\![\Delta]\!] \to [\![g]\!]_k$ (i.h. on 1)
3. $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_{k_0} \to [\![g]\!]_k$ (Weakening on 2)

□

Lemma A.13 (Completeness)**.** *If* $\Sigma; [\![d_1]\!]_{k_1}, \ldots, [\![d_n]\!]_{k_n} \to [\![g]\!]_k$ *is provable in first-order logic, then* $\Sigma; k_1 \mathsf{claims}\ d_1, \ldots, k_n \mathsf{claims}\ d_n \xrightarrow{k} g$ *is provable in* BL_S.

Proof. We perform a lexicographic induction, first on the given derivation of $\Sigma; [\![d_1]\!]_{k_1}, \ldots, [\![d_n]\!]_{k_n} \to [\![g]\!]_k$, and then on the structure of g. We perform a case analysis on the principal formula of the last rule in the derivation. Let Θ denote a hypotheses of the form $[\![d_1]\!]_{k_1}, \ldots, [\![d_n]\!]_{k_n}$, and $|\Theta|$ denote $k_1 \mathsf{claims}\ d_1, \ldots, k_n \mathsf{claims}\ d_n$.

Case. Principal formula of the last rule is atomic. Then the derivation must have the form:

$$\frac{}{\Sigma; \Theta, [\![p]\!]_k \to [\![p']\!]_{k'}}\text{init (because } [\![p]\!]_k = [\![p']\!]_{k'})$$

Since, $(p\ k) = [\![p]\!]_k = [\![p']\!]_{k'} = (p'\ k')$, we must have $p = p'$ and $k = k'$. Therefore we must show that $\Sigma; |\Theta|, k \mathsf{claims}\ p \xrightarrow{k} p$. This follows from rule (claims).

Case. Principal formula of the last rule appears on the left. Hence the last rule must be a left rule, and the principal formula must have the form $[\![d]\!]_{k'}$. Now we case analyze d.

Subcase. $d = p$. This is already covered in the first case.

Subcase. $d = \forall x{:}\sigma.d'$. Then the derivation must have the form:

$$\frac{\Sigma; \Theta, \forall x{:}\sigma.[\![d']\!]_{k'}, [\![d']\!]_{k'}[t/x] \to [\![g]\!]_k \qquad \Sigma \vdash t : \sigma}{\Sigma; \Theta, \forall x{:}\sigma.[\![d']\!]_{k'} \to [\![g]\!]_k}\forall\text{L}$$

To show: $\Sigma; |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d' \xrightarrow{k} g$

1. $\Sigma; |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d', k' \mathsf{claims}\ d'[t/x] \xrightarrow{k} g$ (i.h. on premise)
2. $\Sigma; |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d', \forall x{:}\sigma.d', d'[t/x] \xrightarrow{k'} d'[t/x]$ (Theorem 3.11)
3. $\Sigma; |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d', \forall x{:}\sigma.d' \xrightarrow{k'} d'[t/x]$ (Rule (∀L) on 2)
4. $\Sigma; |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d' \xrightarrow{k'} d'[t/x]$ (Rule (claims) on 3)
5. $\Sigma; (|\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d')| \xrightarrow{k'} d'[t/x]$

   $(4;\ (|\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d')| = |\Theta|, k' \mathsf{claims}\ \forall x{:}\sigma.d')$

6. $\Sigma; |\Theta|, k' \text{ claims } \forall x{:}\sigma.d' \xrightarrow{k} g$ (Theorem 3.10 on 5,1)

Subcase. $d = g' \supset d'$. Then the derivation must have the form:

$$\dfrac{\Sigma; \Theta, [\![g']\!]_{k'} \supset [\![d']\!]_{k'} \rightarrow [\![g']\!]_{k'} \qquad \Sigma; \Theta, [\![g']\!]_{k'} \supset [\![d']\!]_{k'}, [\![d']\!]_{k'} \rightarrow [\![g]\!]_{k}}{\Sigma; \Theta, [\![g']\!]_{k'} \supset [\![d']\!]_{k'} \rightarrow [\![g]\!]_{k}} \supset\text{L}$$

To show: $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d') \xrightarrow{k} g$

1. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d') \xrightarrow{k'} g'$ (i.h. on 1st premise)
2. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d'), k' \text{ claims } d' \xrightarrow{k} g$ (i.h. on 2nd premise)
3. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d'), g' \supset d', d' \xrightarrow{k'} d'$ (Theorem 3.11)
4. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d'), g' \supset d' \xrightarrow{k'} d'$ (Rule ($\supset$L) on 1,3)
5. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d') \xrightarrow{k'} d'$ (Rule (claims) on 4)
6. $\Sigma; (|\Theta|, k' \text{ claims } (g' \supset d'))| \xrightarrow{k'} d'$
   ($5;\ (|\Theta|, k' \text{ claims } (g' \supset d'))| = |\Theta|, k' \text{ claims } (g' \supset d')$)
7. $\Sigma; |\Theta|, k' \text{ claims } (g' \supset d') \xrightarrow{k} g$ (Theorem 3.10 on 6,2)

Subcase. $d = \top$ does not arise since there is no left rule for $\top$.

Subcase. $d = d_1 \wedge d_2$ is similar to the subcase $d = g' \supset d'$.

Case. Principal formula of the last rule appears on the right. Hence the last rule must a right rule, and the principal formula is $[\![g]\!]_k$. We now case analyze the form of g.

Subcase. $g = p$. This is already covered in the first case.

Subcase. $g = k' \text{ says } g'$. Let the given derivation prove $\Sigma; \Theta \rightarrow [\![k' \text{ says } g']\!]_k$. Then we have to show that $\Sigma; |\Theta| \xrightarrow{k} k' \text{ says } g'$.

1. $\Sigma; \Theta \rightarrow [\![g']\!]_{k'}$ (Assumption; $[\![k' \text{ says } g']\!]_k = [\![g']\!]_{k'}$)
2. $\Sigma; |\Theta| \xrightarrow{k'} g'$ (i.h. on 1, smaller g')
3. $\Sigma; (|\Theta|)| \xrightarrow{k'} g'$ ($2;\ (|\Theta|)| = |\Theta|$)
4. $\Sigma; (|\Theta|)| \xrightarrow{k} k' \text{ says } g'$ (Rule (saysR) on 3)

Subcase. $g = g_1 \wedge g_2$. The derivation must have the form:

$$\frac{\Sigma; \Theta \to [\![g_1]\!]_k \qquad \Sigma; \Theta \to [\![g_2]\!]_k}{\Sigma; \Theta \to [\![g_1 \wedge g_2]\!]_k} \wedge\mathrm{R}$$

To show: $\Sigma; |\Theta| \xrightarrow{k} g_1 \wedge g_2$

1. $\Sigma; |\Theta| \xrightarrow{k} g_1$ (i.h. on 1st premise)
2. $\Sigma; |\Theta| \xrightarrow{k} g_2$ (i.h. on 2nd premise)
3. $\Sigma; |\Theta| \xrightarrow{k} g_1 \wedge g_2$ (Rule ($\wedge$R) on 1,2)

Subcases for the remaining forms of g are similar to the subcase $g = g_1 \wedge g_2$.

□

Theorem A.14 (Correctness of Translation; Theorem 3.21)**.** *Let $\Sigma; \Delta, \Xi \xrightarrow{k} g$ be a sequent in the Horn fragment of BL_S and assume that for each $d \in \Xi$, k claims $d \in \Delta$. Then $\Sigma; \Delta, \Xi \xrightarrow{k} g$ is provable in BL_S if and only if its translation $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$ is provable in first-order logic.*

Proof. Suppose $\Sigma; \Delta, \Xi \xrightarrow{k} g$ is provable in BL_S. Then by Lemma A.12, $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$.

Conversely, suppose that $\Sigma; [\![\Delta]\!], [\![\Xi]\!]_k \to [\![g]\!]_k$, and assume that

1. $\Xi = d_1, \ldots, d_n$
2. $\Delta = \{k \text{ claims } d_1, \ldots, k \text{ claims } d_n\} \cup \{k'_1 \text{ claims } d'_1, \ldots, k'_m \text{ claims } d'_m\}$

Then $[\![\Xi]\!]_k = [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k$ and $[\![\Delta]\!] = [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k, [\![d'_1]\!]_{k'_1}, \ldots, [\![d'_m]\!]_{k'_m}$. Hence the given derivation proves:

$$\Sigma; [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k, [\![d'_1]\!]_{k'_1}, \ldots, [\![d'_m]\!]_{k'_m}, [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k \to [\![g]\!]_k$$

By contraction in first-order logic, there must also be a derivation of

$$\Sigma; [\![d_1]\!]_k, \ldots, [\![d_n]\!]_k, [\![d'_1]\!]_{k'_1}, \ldots, [\![d'_m]\!]_{k'_m} \to [\![g]\!]_k$$

Hence by Lemma A.13, there is a BL_S derivation of

$$\Sigma; k \text{ claims } d_1, \ldots, k \text{ claims } d_n, k'_1 \text{ claims } d'_1, \ldots, k'_m \text{ claims } d'_m \xrightarrow{k} g$$

or equivalently, there is a derivation of

$$\Sigma; \Delta \xrightarrow{k} g$$

By weakening (Theorem 3.8), there must also be a derivation of

$$\Sigma; \Delta, \Xi \xrightarrow{k} g$$

□

Appendix B

Proofs from §4

B.1 Proofs from §4.2.3

Lemma B.1 (Constraint substitution)**.** *Suppose the following hold:*

1. $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models c_0$

Then, $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *by a derivation of shorter or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$, and case analysis of its last rule. The interesting cases are rules where $\Sigma; \Psi \models \cdot$ is used in one of the premises. In such cases, we appeal to the assumption (C-cut) from §4.2.1. For the remaining rules, we just apply the induction hypothesis to the premises, and reapply the rule to the modified premises. We show one of the interesting cases here.

Case. $\dfrac{\Sigma; \Psi, c_0 \models u_1' \leq u_1 \qquad \Sigma; \Psi, c_0 \models u_2 \leq u_2'}{\Sigma; \Psi, c_0; E; \Gamma, s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]}\text{hyp}$

1. $\Sigma; \Psi \models c_0$ (Assumption)
2. $\Sigma; \Psi \models u_1' \leq u_1$ ((C-cut) on 1 and 1st premise)
3. $\Sigma; \Psi \models u_2 \leq u_2'$ ((C-cut) on 1 and 2nd premise)
4. $\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \vdash^{\nu} s \circ [u_1, u_2]$ (Rule (hyp) on 2,3)

□

Theorem B.2 (Time subsumption; Theorem 4.4)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$

3. $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m]$

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. Some representative and interesting cases are shown below.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_1, u_2]} \wedge\text{I}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_n, u_m]$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_n, u_m]$ (i.h. on 2nd premise)
3. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \wedge s_2 \circ [u_n, u_m]$ (Rule ($\wedge$I) on 1 and 2)

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \supset\text{I}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)
2. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq u_n$ ((C-weaken) from §4.2.1 on 1)
3. $\Sigma; \Psi, u_n \leq x_1 \models u_n \leq x_1$ ((C-hyp) from §4.2.1)
4. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq x_1$ ((C-trans-time) from §4.2.1 on 2,3)
5. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2]$ (premise)
6. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2]$ (Lemma B.1 on 4,5)
7. $\Sigma; \Psi, x_2 \leq u_m \models x_2 \leq u_2$ (Similar to 4)
8. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_m; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2]$ (Lemma B.1 on 7,6)
9. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_n, u_m]$ (Rule ($\supset$I) on 8)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1, u_2] \qquad \Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1, u_2]} \supset\text{E}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)
2. $\Sigma; \Psi \models u_1' \leq u_n$ ((C-trans-time) from §4.2.1 on 1 and 3rd premise)
3. $\Sigma; \Psi \models u_m \leq u_2$ (Assumption 3)
4. $\Sigma; \Psi \models u_m \leq u_2'$ ((C-trans-time) from §4.2.1 on 3 and 4th premise)
5. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_n, u_m]$ (i.h. on premise)
6. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_n, u_m]$ (Rule ($\supset$E) on 1st premise and 2,4,5)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma| \vdash^{k,u_1,u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } s \circ [u_1, u_2]}\text{saysI}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } s \circ [u_n, u_m]$

1. $\Sigma; \Psi \models k \succeq k$ ((C-refl-prin) from §4.2.1)
2. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)
3. $\Sigma; \Psi \models u_m \leq u_2$ (Assumption 3)
4. $\Sigma; \Psi; E; \Gamma| \vdash^{k,u_n,u_m} s \circ [u_1, u_2]$ (Theorem 4.3 on 1,2,3 and premise)
5. $\Sigma; \Psi; E; \Gamma| \vdash^{k,u_n,u_m} s \circ [u_n, u_m]$ (i.h. on 4)
6. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \text{ says } s \circ [u_n, u_m]$ (Rule (saysI) on 5)

□

B.2 Proofs from §4.2.5

Lemma B.3 (Constraint substitution)**.** *Suppose the following hold:*

1. $\Sigma; \Psi, c_0; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models c_0$

Then, $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *by a derivation of shorter or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi, c_0; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$, and case analysis of its last rule. The interesting cases are rules where $\Sigma; \Psi \models \cdot$ is used in one of the premises. In such cases, we appeal to the assumption (C-cut) from §4.2.1. We show one of the interesting cases here.

Case. $\dfrac{\Sigma; \Psi, c_0 \models u_1' \leq u_1 \qquad \Sigma; \Psi, c_0 \models u_2 \leq u_2'}{\Sigma; \Psi, c_0; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1, u_2]}\text{init}$

1. $\Sigma; \Psi \models c_0$ (Assumption)

2. $\Sigma; \Psi \models u_1' \leq u_1$ ((C-cut) on 1 and 1st premise)
3. $\Sigma; \Psi \models u_2 \leq u_2'$ ((C-cut) on 1 and 2nd premise)
4. $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1, u_2]$ (Rule (hyp) on 2,3)

□

Theorem B.4 (Time subsumption; Theorem 4.11)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$
3. $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_n, u_m]$.

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ and case analysis of its last rule. Some representative and interesting cases are shown below.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_1, u_2]} \wedge\text{R}$

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_n, u_m]$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_n, u_m]$ (i.h. on 2nd premise)
3. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_n, u_m]$ (Rule ($\wedge$R) on 1 and 2)

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \supset\text{R}$

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)
2. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq u_n$ ((C-weaken) from §4.2.1 on 1)
3. $\Sigma; \Psi, u_n \leq x_1 \models u_n \leq x_1$ ((C-hyp) from §4.2.1)
4. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq x_1$ ((C-trans-time) from §4.2.1 on 2,3)
5. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ (premise)
6. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$

(Lemma B.3 on 4,5)

7. $\Sigma; \Psi, x_2 \leq u_m \models x_2 \leq u_2$ (Similar to 4)

8. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_m; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$

(Lemma B.3 on 7,6)

9. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_n, u_m]$ (Rule ($\supset$R) on 8)

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'] \xrightarrow{\nu} s_1 \circ [u_1'', u_2''] \\ \Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'], s_2 \circ [u_1'', u_2''] \xrightarrow{\nu} s \circ [u_1, u_2] \\ \Sigma; \Psi \models u_1' \leq u_1'' \qquad \Sigma; \Psi \models u_2'' \leq u_2'\end{array}}{\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]} \supset\text{L}$

To show: $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_n, u_m]$

1. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'], s_2 \circ [u_1'', u_2''] \xrightarrow{\nu} s \circ [u_n, u_m]$ (i.h. on 2nd premise)

2. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_n, u_m]$

(Rule ($\supset$L) on 1st premise, 1, 3rd premise, 4th premise)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysR}$

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k \ \mathsf{says}\ s \circ [u_n, u_m]$

1. $\Sigma; \Psi \models k \succeq k$ ((C-refl-prin) from §4.2.1)

2. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)

3. $\Sigma; \Psi \models u_m \leq u_2$ (Assumption 3)

4. $\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_n, u_m} s \circ [u_1, u_2]$ (Theorem 4.10 on 1,2,3 and premise)

5. $\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_n, u_m} s \circ [u_n, u_m]$ (i.h. on 5)

6. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k \ \mathsf{says}\ s \circ [u_n, u_m]$ (Rule (saysR) on 6)

□

Lemma B.5 (Interpreted atom substitution)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E, i; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$
2. $\Sigma; E \models i$

Then, $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *by a derivation of shorter or equal depth.*

Proof. By induction on the given derivation of $\Sigma; \Psi; E, i; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$, and case analysis of its last rule. The only interesting case is rule (interR) where we appeal to the assumption (S-cut) from §4.2.1. This case is shown below.

Case. $\dfrac{\Sigma; E, i \models i'}{\Sigma; \Psi; E, i; \Gamma \xrightarrow{\nu} i' \circ [u_1, u_2]}\text{interR}$

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} i' \circ [u_1, u_2]$

1. $\Sigma; E \models i'$ ((S-cut) from §4.2.1 on premise and assumption 2)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} i' \circ [u_1, u_2]$ (Rule (interR) on 1)

□

Theorem B.6 (Admissibility of cut; Theorem 4.12)**.** *The following two properties hold:*

1. *Suppose that*
   (a) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *and*
   (b) $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$

   Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$.
2. *Suppose that*
   (a) $\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]$
   (b) $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$

   Then $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$.

Proof. Both (1) and (2) are proved by a simultaneous lexicographic induction, first on the size of the cut formula s, then on the order (2) > (1) on the hypotheses, and then on the depths of the two given derivations. Let $\mathcal{D}$ denote the derivation in (a) and let $\mathcal{E}$ denote the derivation in (b).

Proof of (1). For proving (1) we case analyze the last rules in $\mathcal{D}$ and $\mathcal{E}$ and distinguish four sets of cases, in addition to the special case where $\mathcal{E}$ ends in (init): (A) $\mathcal{D}$ ends in a left rule, (B) $\mathcal{E}$ ends in a right rule, (C) $\mathcal{E}$ ends in a left rule but the judgment being cut is not principal in the rule, and (D) $\mathcal{E}$ ends in a left rule, $\mathcal{D}$ ends in a right rule and the cut judgment is principal. The reader may easily check that these sets of cases are exhaustive.[1] The cases in (A), (B), and (C) are straightforward. We show here the cases where $\mathcal{E}$ ends in (init), and some of the cases in (D).

[1] Note that we do not need to explicitly consider the case where $\mathcal{D}$ ends in rule (init). This is a consequence of restricting the (init) rule to uninterpreted atoms. If we were to generalize the (init) rule to arbitrary formulas, then an explicit consideration of this case would be necessary.

Case. $\mathcal{E} = \dfrac{\Sigma; \Psi \models u_1' \leq u_1'' \qquad \Sigma; \Psi \models u_2'' \leq u_2'}{\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1'', u_2'']}\text{init}$

Subcase. The cut judgment is not $p \circ [u_1', u_2']$. So let $\Gamma = \Gamma', s \circ [u_1, u_2]$ and let the judgment being cut be $s \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma', p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1'', u_2'']$. This follows by rule (init) on the premises of the given derivation.

Subcase. The cut judgment is $p \circ [u_1', u_2']$. So $\mathcal{D}$ proves $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} p \circ [u_1', u_2']$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} p \circ [u_1'', u_2'']$. This follows by Theorem 4.11 on the derivation $\mathcal{D}$.

Case. $\mathcal{D} = \dfrac{\Sigma; \Psi; E; \Gamma | \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k \text{ says } s \circ [u_1, u_2]}\text{saysR}$

$\mathcal{E} = \dfrac{\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2], k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{saysL}$

And the cut judgment is $k \text{ says } s \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$.

1. $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(1) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(2) on premise of $\mathcal{D}$ and 1)

The use of the i.h. in the second step is justified because the cut formula s is strictly smaller than the cut formula $k \text{ says } s$ that we started with.

Case. $\mathcal{D} = \dfrac{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]}@\text{R}$

$\mathcal{E} = \dfrac{\Sigma; \Psi; E; \Gamma, s \mathbin{@} [u_1', u_2'] \circ [u_1, u_2], s \circ [u_1', u_2'] \xrightarrow{\nu} r \circ [u_1'', u_2'']}{\Sigma; \Psi; E; \Gamma, s \mathbin{@} [u_1', u_2'] \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1'', u_2'']}@\text{L}$

And the cut judgment is $s \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1'', u_2'']$.

1. $\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \xrightarrow{\nu} r \circ [u_1'', u_2'']$ (i.h.(1) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1'', u_2'']$ (i.h.(1) on premise of $\mathcal{D}$ and 1)

Case. $\mathcal{D} = \dfrac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} c \circ [u_1, u_2]}\text{consR}$

$\mathcal{E} = \dfrac{\Sigma; \Psi, c; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{consL}$

And the cut judgment is $c \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$.

1. $\Sigma; \Psi, c; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(1) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (Lemma B.3 on premise of $\mathcal{D}$ and 1)

Case. $\mathcal{D} = \dfrac{\Sigma; E \models i}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} i \circ [u_1, u_2]}\text{interR}$

$\mathcal{E} = \dfrac{\Sigma; \Psi; E, i; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{interL}$

And the cut judgment is $i \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$.

1. $\Sigma; \Psi; E, i; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(1) on $\mathcal{D}$ and premise of $\mathcal{E}$)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (Lemma B.5 on premise of $\mathcal{D}$ and 1)

Proof of (2). To prove (2) we case analyze the last rule in $\mathcal{E}$. There is only one interesting case, which we show here.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2], s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2'] \quad \nu = k', u_b, u_e \quad \Sigma; \Psi \models u_1 \leq u_b \quad \Sigma; \Psi \models u_e \leq u_2 \quad \Sigma; \Psi \models k \succeq k'}{\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{claims}$

$\mathcal{D}$ proves $\Sigma; \Psi; E; \Gamma| \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]$. To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$.

1. $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(2) on $\mathcal{D}$ and 1st premise of $\mathcal{E}$)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]$ (Weakening Theorem 4.8 on $\mathcal{D}$)
3. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ (Theorem 4.10 on 2)
4. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ (i.h.(1) on 3,1)

Use of the i.h. in the last step is justified because we assume the ordering (2) > (1) among the inductive hypotheses. □

Theorem B.7 (Equivalence; Theorem 4.14)**.** *The following are equivalent.*

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *in the sequent calculus.*
2. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *in natural deduction.*

Proof. We prove separately that (1) ⇒ (2) and (2) ⇒ (1).

Proof that (1) ⇒ (2). By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$, and case analysis of its last rule. The cases where the derivation ends in a right rule (or the rule (init)) are uninteresting – we apply the i.h. to sequents in the premises and use the corresponding introduction rule (or the rule (hyp)) in natural deduction. We show here the case of the rule (claims) and some of the interesting left rules.

Case. $\dfrac{\begin{array}{c}\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2],s\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2'] \\ \nu=k',u_b,u_e \qquad \Sigma;\Psi\models u_1\le u_b \qquad \Sigma;\Psi\models u_e\le u_2 \qquad \Sigma;\Psi\models k\succeq k'\end{array}}{\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2']}\mathsf{claims}$

To show: $\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$

1. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2],s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (i.h. on 1st premise)
2. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu} s\circ[u_1,u_2]$ (Rule (claims) in natural deduction)
3. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (Theorem 4.5 on 2,1)

Case. $\dfrac{\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2],k\ \mathsf{claims}\ s\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2']}{\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2']}\mathsf{saysL}$

To show: $\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$

1. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2]\vdash^{\nu} k\ \mathsf{says}\ s\circ[u_1,u_2]$ (Rule (hyp))
2. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2],k\ \mathsf{claims}\ s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (i.h. on premise)
3. $\Sigma;\Psi;E;\Gamma,k\ \mathsf{says}\ s\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (Rule (saysE) on 1,2)

Case. $\dfrac{\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2],s\circ[u_1',u_2']\xrightarrow{\nu} r\circ[u_1'',u_2'']}{\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1'',u_2'']}@\mathrm{L}$

To show: $\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1'',u_2'']$

1. $\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2]\vdash^{\nu} s\ @\ [u_1',u_2']\circ[u_1,u_2]$ (Rule (hyp))
2. $\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2],s\circ[u_1',u_2']\vdash^{\nu} r\circ[u_1'',u_2'']$ (i.h. on premise)
3. $\Sigma;\Psi;E;\Gamma,s\ @\ [u_1',u_2']\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1'',u_2'']$ (Rule (@E) on 1,2)

Case. $\dfrac{\Sigma;\Psi,c;E;\Gamma,c\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2']}{\Sigma;\Psi;E;\Gamma,c\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1',u_2']}\mathrm{consL}$

To show: $\Sigma;\Psi;E;\Gamma,c\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$

1. $\Sigma;\Psi;E;\Gamma,c\circ[u_1,u_2]\vdash^{\nu} c\circ[u_1,u_2]$ (Rule (hyp))
2. $\Sigma;\Psi,c;E;\Gamma,c\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (i.h. on premise)
3. $\Sigma;\Psi;E;\Gamma,c\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1',u_2']$ (Rule (consE) on 1,2)

Case. $\dfrac{\begin{array}{c}\Sigma;\Psi;E;\Gamma,s_1\supset s_2\circ[u_1,u_2]\xrightarrow{\nu} s_1\circ[u_1',u_2'] \\ \Sigma;\Psi;E;\Gamma,s_1\supset s_2\circ[u_1,u_2],s_2\circ[u_1',u_2']\xrightarrow{\nu} r\circ[u_1'',u_2''] \\ \Sigma;\Psi\models u_1\le u_1' \qquad \Sigma;\Psi\models u_2'\le u_2\end{array}}{\Sigma;\Psi;E;\Gamma,s_1\supset s_2\circ[u_1,u_2]\xrightarrow{\nu} r\circ[u_1'',u_2'']}\supset\mathrm{L}$

To show: $\Sigma;\Psi;E;\Gamma,s_1\supset s_2\circ[u_1,u_2]\vdash^{\nu} r\circ[u_1'',u_2'']$

1. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^\nu s_1 \supset s_2 \circ [u_1, u_2]$ (Rule (hyp))
2. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^\nu s_1 \circ [u_1', u_2']$ (i.h. on 1st premise)
3. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^\nu s_2 \circ [u_1', u_2']$

   (Rule ($\supset$E) on 1,2 and 3rd,4th premise)
4. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2], s_2 \circ [u_1', u_2'] \vdash^\nu r \circ [u_1'', u_2'']$ (i.h. on 2nd premise)
5. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^\nu r \circ [u_1'', u_2'']$ (Theorem 4.5 on 3,4)

Proof that (2) $\Rightarrow$ (1). By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^\nu s \circ [u_1, u_2]$, and case analysis of its last rule. The cases where the derivation ends in an introduction rule are uninteresting – we apply the i.h. to hypothetical judgments in the premises and use the corresponding right rule in the sequent calculus. We show here the case of the rules (hyp), (claims), and some of the interesting elimination rules.

Case. $$\frac{\Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \vdash^\nu s \circ [u_1, u_2]}\text{hyp}$$

To show: $\Sigma; \Psi; E; \Gamma, s \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]$. This follows from Theorem 4.13 applied to the premises of the rule.

Case. $$\frac{\begin{array}{c}\nu = k, u_b, u_e \qquad \Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2' \\ \Sigma; \Psi \models u_1' \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2' \qquad \Sigma; \Psi \models k' \succeq k\end{array}}{\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'] \vdash^\nu s \circ [u_1, u_2]}\text{claims}$$

To show: $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'], s \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]$

   (Theorem 4.13 on 2nd, 3rd premises)
2. $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1', u_2'] \xrightarrow{\nu} s \circ [u_1, u_2]$

   (Rule (claims) on 1 and 4th,5th,6th premises)

Case. $$\frac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^\nu k \ \mathsf{says}\ s \circ [u_1, u_2] \\ \Sigma; \Psi; E; \Gamma, k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^\nu s' \circ [u_1', u_2']\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^\nu s' \circ [u_1', u_2']}\mathsf{saysE}$$

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k \ \mathsf{says}\ s \circ [u_1, u_2]$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Gamma, k \ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$ (i.h. on 2nd premise)
3. $\Sigma; \Psi; E; \Gamma, k \ \mathsf{says}\ s \circ [u_1, u_2], k \ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$

(Weakening Theorem 4.8 on 2)

4. $\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$ (Rule (saysL) on 3)
5. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$ (Theorem 4.12 on 1,4)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \mathbin{@} [u_1, u_2] \circ [u_1', u_2'] \qquad \Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \vdash^{\nu} s' \circ [u_1'', u_2'']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1'', u_2'']}$@E

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1'', u_2'']$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \mathbin{@} [u_1, u_2] \circ [u_1', u_2']$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1'', u_2'']$ (i.h. on 2nd premise)
3. $\Sigma; \Psi; E; \Gamma, s \mathbin{@} [u_1, u_2] \circ [u_1', u_2'], s \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1'', u_2'']$

(Weakening Theorem 4.8 on 2)

4. $\Sigma; \Psi; E; \Gamma, s \mathbin{@} [u_1, u_2] \circ [u_1', u_2'] \xrightarrow{\nu} s' \circ [u_1'', u_2'']$ (Rule (@L) on 3)
5. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1'', u_2'']$ (Theorem 4.12 on 1,4)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} c \circ [u_1, u_2] \qquad \Sigma; \Psi, c; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s' \circ [u_1', u_2']}$consE

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} c \circ [u_1, u_2]$ (i.h. on 1st premise)
2. $\Sigma; \Psi, c; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$ (i.h. on 2nd premise)
3. $\Sigma; \Psi, c; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$ (Weakening Theorem 4.8 on 2)
4. $\Sigma; \Psi; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{\nu} s' \circ [u_1', u_2']$ (Rule (consL) on 3)
5. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s' \circ [u_1', u_2']$ (Theorem 4.12 on 1,4)

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \\ \Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \circ [u_1', u_2'] \qquad \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_2 \circ [u_1', u_2']}$⊃E

To show: $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2], s_2 \circ [u_1', u_2'] \xrightarrow{\nu} s_2 \circ [u_1', u_2']$ (Theorem 4.13)
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1', u_2']$ (i.h. on 2nd premise)
3. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} s_1 \circ [u_1', u_2']$ (Weakening Theorem 4.8 on 2)
4. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} s_2 \circ [u_1', u_2']$

(Rule ($\supset$L) on 1,3 and 3rd,4th premises)

5. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2]$ (i.h. on 1st premise)
6. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1', u_2']$ (Theorem 4.12 on 5,4)

□

B.3 Proofs from §4.5

Lemma B.8 (Constraint substitution)**.** *Suppose* $\Sigma; \Psi \models c_0$*. Then the following hold.*

1. $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ *by a derivation of shorter or equal depth.*
2. $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ *by a derivation of shorter or equal depth.*

Proof. By simultaneous induction on derivations given in (1) and (2) and case analysis of their last rules. The interesting cases are rules where $\Sigma; \Psi \models \cdot$ is used in one of the premises. In such cases, we appeal to the assumption (C-cut) from §4.2.1. For the remaining rules, we just apply the induction hypothesis to the premises, and reapply the rule to the modified premises. We show one of the interesting cases here.

Case.
$$\frac{\nu = k, u_b, u_e \qquad \Sigma; \Psi, c_0 \models u_1 \leq u_b \qquad \Sigma; \Psi, c_0 \models u_e \leq u_2 \qquad \Sigma; \Psi, c_0 \models k' \succeq k}{\Sigma; \Psi, c_0; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow}\text{claims}$$

To show: $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$

1. $\Sigma; \Psi \models c_0$ (Assumption)
2. $\Sigma; \Psi \models u_1 \leq u_b$ ((C-cut) on 1 and 2nd premise)
3. $\Sigma; \Psi \models u_e \leq u_2$ ((C-cut) on 1 and 3rd premise)
4. $\Sigma; \Psi \models k' \succeq k$ ((C-cut) on 1 and 4th premise)
5. $\Sigma; \Psi; E; \Gamma, k' \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$ (Rule (claims) on 2,3,4)

□

Theorem B.9 (Time subsumption; Theorem 4.17)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$
2. $\Sigma; \Psi \models u_1 \leq u_n$ *and* $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m] \Uparrow$

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$ and case analysis of its last rule. Some interesting cases are shown below.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1', u_2'] \Downarrow \qquad \Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow} \Downarrow\Uparrow$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m] \Uparrow$

1. $\Sigma; \Psi \models u_1' \leq u_1$ (premise)
2. $\Sigma; \Psi \models u_1 \leq u_n$ (assumption)
3. $\Sigma; \Psi \models u_1' \leq u_n$ ((C-trans-time) from §4.2.1 on 1,2)
4. $\Sigma; \Psi \models u_m \leq u_2'$ (Similar to 3)
5. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_n, u_m] \Uparrow$ (Rule ($\Downarrow\Uparrow$) on 1st premise,3,4)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_1, u_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says} \ s \circ [u_1, u_2] \Uparrow} \mathsf{saysI}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says} \ s \circ [u_n, u_m] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma| \vdash^{k, u_1, u_2} s \circ [u_n, u_m] \Uparrow$ (i.h. on premise)
2. $\Sigma; \Psi; E; \Gamma| \vdash^{k, u_n, u_m} s \circ [u_n, u_m] \Uparrow$ (Theorem 4.16 on 1)
3. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k \ \mathsf{says} \ s \circ [u_n, u_m] \Uparrow$ (Rule (saysI) on 2)

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2] \Uparrow}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \Uparrow} \supset\mathrm{I}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_n, u_m] \Uparrow$

1. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)
2. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq u_n$ ((C-weaken) from §4.2.1 on 1)
3. $\Sigma; \Psi, u_n \leq x_1 \models u_n \leq x_1$ ((C-hyp) from §4.2.1)
4. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq x_1$ ((C-trans-time) from §4.2.1 on 2,3)
5. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2] \Uparrow$ (premise)
6. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2] \Uparrow$ (Lemma B.8 on 4,5)
7. $\Sigma; \Psi, x_2 \leq u_m \models x_2 \leq u_2$ (Similar to 4)

8. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \le x_1, x_2 \le u_m; E; \Gamma, s_1 \circ [x_1, x_2] \vdash^{\nu} s_2 \circ [x_1, x_2] \Uparrow$

(Lemma B.8 on 7,6)

9. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s_1 \supset s_2 \circ [u_n, u_m] \Uparrow$ (Rule ($\supset$I) on 8)

□

Theorem B.10 (Normalization; Theorem 4.19)**.** *Suppose* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2]$ *in natural deduction. Then* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$.

Proof. By Theorem 4.14 it suffices to show that $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ implies $\Sigma; \Psi; E; \Gamma \vdash^{\nu} s \circ [u_1, u_2] \Uparrow$. We prove the latter by induction on the depth of the given sequent calculus proof and a case analysis of its last rule. Some representative cases are shown below.

Case. $\dfrac{\Sigma; \Psi \models u_1' \le u_1 \qquad \Sigma; \Psi \models u_2 \le u_2'}{\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1, u_2]}\text{init}$

To show: $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \vdash^{\nu} p \circ [u_1, u_2] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \vdash^{\nu} p \circ [u_1', u_2'] \Downarrow$ (Rule (hyp))
2. $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \vdash^{\nu} p \circ [u_1, u_2] \Uparrow$ (Rule ($\Downarrow\Uparrow$) on 1 and the premises)

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2], s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2'] \\ \nu = k', u_b, u_e \qquad \Sigma; \Psi \models u_1 \le u_b \qquad \Sigma; \Psi \models u_e \le u_2 \qquad \Sigma; \Psi \models k \succeq k'\end{array}}{\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{claims}$

To show: $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} s \circ [u_1, u_2] \Downarrow$

(Rule (claims) on 3rd,4th,5th premises)

2. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2], s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$ (i.h. on 1st premise)
3. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$ (Theorem 4.18 on 1,2)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \mid \xrightarrow{k, u_1, u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysR}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma \mid \vdash^{k, u_1, u_2} s \circ [u_1, u_2] \Uparrow$ (i.h. on premise)
2. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2] \Uparrow$ (Rule (saysI) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2], k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\mathsf{saysL}$

To show: $\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2] \vdash^{\nu} k \text{ says } s \circ [u_1, u_2] \Downarrow$ (Rule (hyp))
2. $\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2], k \text{ claims } s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$ (i.h. on premise)
3. $\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1', u_2'] \Uparrow$ (Rule (saysE) on 1,2)

Case. $$\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} s_1 \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2], s_2 \circ [u_1', u_2'] \xrightarrow{\nu} r \circ [u_1'', u_2''] \\ \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1'', u_2'']} \supset\text{L}$$

To show: $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1'', u_2''] \Uparrow$

1. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} s_1 \supset s_2 \circ [u_1, u_2] \Downarrow$ (Rule (hyp))
2. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} s_1 \circ [u_1', u_2'] \Uparrow$ (i.h. on 1st premise)
3. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} s_2 \circ [u_1', u_2'] \Uparrow$

   (Rule ($\supset$E) on 1,2 and 3rd,4th premises)
4. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2], s_2 \circ [u_1', u_2'] \vdash^{\nu} r \circ [u_1'', u_2''] \Uparrow$ (i.h. on 2nd premise)
5. $\Sigma; \Psi; E; \Gamma, s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} r \circ [u_1'', u_2''] \Uparrow$ (Theorem 4.18 on 3,4)

□

B.4 Proofs from §4.6

Theorem B.11 (Correctness of embedding; Theorem 4.21)**.** *Suppose that for every k, k', Σ', and Ψ' not containing $\succeq$, $\Sigma'; \Psi' \models k \succeq k'$ in BL if and only if $\Sigma' \vdash k \succeq k'$ in BL_S. Then, $\Sigma; \Gamma \xrightarrow{k_0} s$ is provable in BL_S if and only if $\ulcorner \Sigma; \Gamma \xrightarrow{k_0} s \urcorner$ is provable in BL.*

Proof. The "if" direction was proved in the main body of the paper. The "only if" direction follows by an induction on the depth of the given BL_S derivation of $\Sigma; \Gamma \xrightarrow{k_0} s$ and a case analysis of its last rule. We show some of the interesting cases here.

Case. $$\dfrac{\Sigma \vdash k \succeq k_0 \qquad \Sigma; \Gamma, k \text{ claims } s, s \xrightarrow{k_0} r}{\Sigma; \Gamma, k \text{ claims } s \xrightarrow{k_0} r} \text{claims}$$

To show: $\Sigma; \cdot; \cdot; \ulcorner \Gamma \urcorner, k \text{ claims } \ulcorner s \urcorner \circ [-\infty, +\infty] \xrightarrow{k_0, -\infty, +\infty} \ulcorner r \urcorner \circ [-\infty, +\infty]$

1. $\Sigma; \cdot; \cdot; \ulcorner \Gamma \urcorner, k \text{ claims } \ulcorner s \urcorner \circ [-\infty, +\infty], \ulcorner s \urcorner \circ [-\infty, +\infty] \xrightarrow{k_0, -\infty, +\infty} \ulcorner r \urcorner \circ [-\infty, +\infty]$

   (i.h. on premise)
2. $\Sigma; \cdot; \cdot; \ulcorner \Gamma \urcorner, k \text{ claims } \ulcorner s \urcorner \circ [-\infty, +\infty] \xrightarrow{k_0, -\infty, +\infty} \ulcorner r \urcorner \circ [-\infty, +\infty]$

(Rule (claims) on 1)

Case. $\dfrac{\Sigma;\Gamma, s \xrightarrow{k} s'}{\Sigma;\Gamma \xrightarrow{k} s \supset s'}\supset\text{R}$

To show: $\Sigma;\cdot;\cdot;\ulcorner\Gamma\urcorner \xrightarrow{\ulcorner k\urcorner} (\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty]$

1. $\Sigma;\cdot;\cdot;\ulcorner\Gamma\urcorner,\ulcorner s\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner s'\urcorner \circ [-\infty,+\infty]$ (i.h. on premise)
2. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; -\infty \le x_1, x_2 \le +\infty;\cdot;\ulcorner\Gamma\urcorner,\ulcorner s\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner s'\urcorner \circ [-\infty,+\infty]$

   (Weakening Theorem 4.8 on 1)
3. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; -\infty \le x_1, x_2 \le +\infty;\cdot;\ulcorner\Gamma\urcorner,\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty] \circ [x_1,x_2] \xrightarrow{\ulcorner k\urcorner} \ulcorner s'\urcorner \circ [-\infty,+\infty]$ (Rule (@L) on 2)
4. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; -\infty \le x_1, x_2 \le +\infty;\cdot;\ulcorner\Gamma\urcorner,\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty] \circ [x_1,x_2] \xrightarrow{\ulcorner k\urcorner} \ulcorner s'\urcorner \circ [x_1,x_2]$ (Theorem 4.11 on 3)
5. $\Sigma;\cdot;\cdot;\ulcorner\Gamma\urcorner \xrightarrow{\ulcorner k\urcorner} (\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty]$ (Rule ($\supset$R) on 4)

Case. $\dfrac{\Sigma;\Gamma, s \supset s' \xrightarrow{k} s \qquad \Sigma;\Gamma, s \supset s', s' \xrightarrow{k} r}{\Sigma;\Gamma, s \supset s' \xrightarrow{k} r}\supset\text{L}$

To show: $\Sigma;\ulcorner\Gamma\urcorner,(\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner r\urcorner \circ [-\infty,+\infty]$

1. $\Sigma;\ulcorner\Gamma\urcorner,(\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner s\urcorner \circ [-\infty,+\infty]$

   (i.h. on 1st premise)
2. $\Sigma;\ulcorner\Gamma\urcorner,(\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner s\urcorner \mathbin{@} [-\infty,+\infty] \circ [-\infty,+\infty]$

   (Rule (@R) on 1)
3. $\Sigma;\ulcorner\Gamma\urcorner,(\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty], s' \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner r\urcorner \circ [-\infty,+\infty]$

   (i.h. on 2nd premise)
4. $\Sigma;\ulcorner\Gamma\urcorner,(\ulcorner s\urcorner \mathbin{@} [-\infty,+\infty]) \supset \ulcorner s'\urcorner \circ [-\infty,+\infty] \xrightarrow{\ulcorner k\urcorner} \ulcorner r\urcorner \circ [-\infty,+\infty]$

   (Rule ($\supset$L) 2,3)

Case. $\dfrac{\Sigma;\Gamma| \xrightarrow{k} s}{\Sigma;\Gamma \xrightarrow{k_0} k \mathsf{\ says\ } s}\mathsf{saysR}$

To show: $\Sigma;\ulcorner\Gamma\urcorner \xrightarrow{\ulcorner k_0\urcorner} k \mathsf{\ says\ } \ulcorner s\urcorner \circ [-\infty,+\infty]$

1. $\Sigma; \ulcorner \Gamma | \urcorner \xrightarrow{k,-\infty,+\infty} \ulcorner s \urcorner \circ [-\infty, +\infty]$ (i.h. on premise)
2. $\ulcorner \Gamma | \urcorner = \ulcorner \Gamma \urcorner |$ (Definition)
3. $\Sigma; \ulcorner \Gamma \urcorner | \xrightarrow{k,-\infty,+\infty} \ulcorner s \urcorner \circ [-\infty, +\infty]$ (1,2)
4. $\Sigma; \ulcorner \Gamma \urcorner \xrightarrow{\ulcorner k_0 \urcorner} k \ \mathsf{says} \ \ulcorner s \urcorner \circ [-\infty, +\infty]$ (Rule (saysR) on 3)

□

Appendix C

Proofs from §5

C.1 Proofs from §5.1.2

Lemma C.1 (Constraint substitution)**.** *Suppose* $\Sigma; \Psi \models c_0$*. Then the following hold.*

1. $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ *by a derivation of shorter or equal depth.*
2. $\Sigma; \Psi, c_0; E; \Gamma \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2]$ *by a derivation of shorter or equal depth.*

Proof. By simultaneous induction on derivations given in (1) and (2) and case analysis of their last rules, as in the proof of Lemma B.8. □

Theorem C.2 (Time subsumption; Theorem 5.5)**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$
2. $\Sigma; \Psi \models u_1 \leq u_n$ *and* $\Sigma; \Psi \models u_m \leq u_2$

Then $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_n, u_m]$

Proof. By induction on the depth of the given derivation of $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$ and case analysis of its last rule. Some interesting cases are shown below.

Case. $\dfrac{\Sigma; \Psi; E; \Gamma \vdash^{\nu} R \Longrightarrow s \circ [u_1', u_2'] \qquad \Sigma; \Psi \models u_1' \leq u_1 \qquad \Sigma; \Psi \models u_2 \leq u_2'}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} R \Longleftarrow s \circ [u_1, u_2]}\text{infer}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} R \Longleftarrow s \circ [u_n, u_m]$

1. $\Sigma; \Psi \models u_1' \leq u_1$ (premise)
2. $\Sigma; \Psi \models u_1 \leq u_n$ (assumption)
3. $\Sigma; \Psi \models u_1' \leq u_n$ ((C-trans-time) from §4.2.1 on 1,2)

4. $\Sigma; \Psi \models u_m \leq u_2'$ (Similar to 3)

5. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} R \Longleftarrow s \circ [u_n, u_m]$ (Rule (infer) on 1st premise,3,4)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma| \vdash^{k,u_1,u_2} V \Longleftarrow s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{saysI}\ V \Longleftarrow k\ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysI}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{saysI}\ V \Longleftarrow k\ \mathsf{says}\ s \circ [u_n, u_m]$

1. $\Sigma; \Psi; E; \Gamma| \vdash^{k,u_1,u_2} V \Longleftarrow s \circ [u_n, u_m]$ (i.h. on premise)

2. $\Sigma; \Psi; E; \Gamma| \vdash^{k,u_n,u_m} V \Longleftarrow s \circ [u_n, u_m]$ (Theorem 5.4 on 1)

3. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{saysI}\ V \Longleftarrow k\ \mathsf{says}\ s \circ [u_n, u_m]$ (Rule (saysI) on 2)

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, \pi : s_1 \circ [x_1, x_2] \vdash^{\nu} V \Longleftarrow s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1 \supset s_2 \circ [u_1, u_2]}\supset\mathrm{I}$

To show: $\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1 \supset s_2 \circ [u_n, u_m]$

1. $\Sigma; \Psi \models u_1 \leq u_n$ (Assumption 2)

2. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq u_n$ ((C-weaken) from §4.2.1 on 1)

3. $\Sigma; \Psi, u_n \leq x_1 \models u_n \leq x_1$ ((C-hyp) from §4.2.1)

4. $\Sigma; \Psi, u_n \leq x_1 \models u_1 \leq x_1$ ((C-trans-time) from §4.2.1 on 2,3)

5. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, \pi : s_1 \circ [x_1, x_2] \vdash^{\nu} V \Longleftarrow s_2 \circ [x_1, x_2]$

(premise)

6. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_2; E; \Gamma, \pi : s_1 \circ [x_1, x_2] \vdash^{\nu} V \Longleftarrow s_2 \circ [x_1, x_2]$

(Lemma C.1 on 4,5)

7. $\Sigma; \Psi, x_2 \leq u_m \models x_2 \leq u_2$ (Similar to 4)

8. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_n \leq x_1, x_2 \leq u_m; E; \Gamma, \pi : s_1 \circ [x_1, x_2] \vdash^{\nu} V \Longleftarrow s_2 \circ [x_1, x_2]$

(Lemma C.1 on 7,6)

9. $\Sigma; \Psi; E; \Gamma \vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1 \supset s_2 \circ [u_n, u_m]$ (Rule ($\supset$I) on 8)

□

C.2 Proofs from §5.2.2

Lemma C.3 (Soundness; Lemma 5.10)**.** *Suppose that the following hold for some $\mathcal{C}$, $\mathcal{I}$, list $\vec{x}$ of term variables, list $\vec{\sigma}$ of sorts, list $\vec{t_0}$ of terms satisfying $\Sigma \vdash \vec{t_0} : \vec{\sigma}$, and system state E_0 not containing any element of $\vec{x}$.*

1. *For each $(\Sigma', \vec{x} : \vec{\sigma}; \Psi' \models c') \in \mathcal{C}$, it is the case that $\Sigma'; \Psi'[\vec{t_0}/\vec{x}] \models c'[\vec{t_0}/\vec{x}]$.*
2. *For each $(\Sigma', \vec{x} : \vec{\sigma}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E_0, E'[\vec{t_0}/\vec{x}] \models i'[\vec{t_0}/\vec{x}]$.*

Then,

A. $\Sigma, \vec{x} : \vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *implies* $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V[\vec{t_0}/\vec{x}] \Longleftarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

B. $\Sigma, \vec{x} : \vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ *implies* $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longrightarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

Proof. By simultaneous induction on the derivations given in A and B and case analysis of their last rules. We show some representative cases.

Case. $\dfrac{\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}'; \mathcal{I} \quad \mathcal{C} = (\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_1 \leq u_1'), (\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_2' \leq u_2)}{\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longleftarrow s \circ [u_1', u_2'] \searrow \mathcal{C}', \mathtt{unsat}(\mathcal{C}); \mathcal{I}}\text{infer}$

To show: $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longleftarrow s[\vec{t_0}/\vec{x}] \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]$

1. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longrightarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$ (i.h. on premise)
2. Case analysis on whether $(\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_1 \leq u_1') \in \mathtt{unsat}(\mathcal{C})$ or not.
   - Case: $(\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_1 \leq u_1') \in \mathtt{unsat}(\mathcal{C})$
     (a) $\Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_1[\vec{t_0}/\vec{x}] \leq u_1'[\vec{t_0}/\vec{x}]$ (Assumption 1)
   - Case: $(\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_1 \leq u_1') \notin \mathtt{unsat}(\mathcal{C})$
     (a) $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi \models u_1 \leq u_1'$ (Defn. of `unsat`)
     (b) $\Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_1[\vec{t_0}/\vec{x}] \leq u_1'[\vec{t_0}/\vec{x}]$ ((C-subst) from §4.2.1)

   $\Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_1[\vec{t_0}/\vec{x}] \leq u_1'[\vec{t_0}/\vec{x}]$ (From case analysis)
3. $\Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_2'[\vec{t_0}/\vec{x}] \leq u_2[\vec{t_0}/\vec{x}]$ (Similar to 2)
4. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longleftarrow s[\vec{t_0}/\vec{x}] \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]$ (Rule (infer) on 1,2,3)

Case. $\dfrac{\begin{array}{c}\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow c \circ [u_1, u_2] \searrow \mathcal{C}_1; \mathcal{I}_1 \\ \Sigma, \vec{x}{:}\vec{\sigma}; \Psi, c; E; \Pi \vdash^{\nu} V \Longleftarrow s' \circ [u_1', u_2'] \searrow \mathcal{C}_2; \mathcal{I}_2\end{array}}{\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} \texttt{consE}\ R\ V \Longleftarrow s' \circ [u_1', u_2'] \searrow \mathcal{C}_1, \mathcal{C}_2; \mathcal{I}_1, \mathcal{I}_2}\text{consE}$

To show: $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{consE}\ (R[\vec{t_0}/\vec{x}])\ (V[\vec{t_0}/\vec{x}]) \Longleftarrow s'[\vec{t_0}/\vec{x}] \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]$

1. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R[\vec{t_0}/\vec{x}] \Longrightarrow c[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$ (i.h. on 1st premise)
2. $\Sigma; \Psi[\vec{t_0}/\vec{x}], c[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V[\vec{t_0}/\vec{x}] \Longleftarrow s'[\vec{t_0}/\vec{x}] \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]$ (i.h. on 2nd premise)
3. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{consE}\ (R[\vec{t_0}/\vec{x}])\ (V[\vec{t_0}/\vec{x}]) \Longleftarrow s'[\vec{t_0}/\vec{x}] \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]$ (Rule (consE) on 1,2)

Case. $\dfrac{\mathcal{I} = (\Sigma, \vec{x}{:}\vec{\sigma}; E \models i)}{\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} \texttt{interI} \Longleftarrow i \circ [u_1, u_2] \searrow \cdot; \texttt{unsat}(\mathcal{I})}\text{interI}$

To show: $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{interI} \Longleftarrow i[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

1. Case analysis on whether $(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i) \in \texttt{unsat}(\mathcal{I})$ or not.
   - Case: $(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i) \in \texttt{unsat}(\mathcal{I})$
     (a) $\Sigma; E_0, E[\vec{t_0}/\vec{x}] \models i[\vec{t_0}/\vec{x}]$ (Assumption 2)
   - Case: $(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i) \notin \texttt{unsat}(\mathcal{I})$
     (a) $\Sigma, \vec{x}{:}\vec{\sigma}; E \models i$ (Defn. of unsat)
     (b) $\Sigma, \vec{x}{:}\vec{\sigma}; E_0, E \models i$ ((S-weaken) from §4.2.1 on a)
     (c) $\Sigma; E_0[\vec{t_0}/\vec{x}], E[\vec{t_0}/\vec{x}] \models i[\vec{t_0}/\vec{x}]$ ((S-subst) from §4.2.1 on b)
     (d) $E_0 = E_0[\vec{t_0}/\vec{x}]$ (Assumption)
     (e) $\Sigma; E_0, E[\vec{t_0}/\vec{x}] \models i[\vec{t_0}/\vec{x}]$ (c,d)

   $\Sigma; E_0, E[\vec{t_0}/\vec{x}] \models i[\vec{t_0}/\vec{x}]$ (From case analysis)
2. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{interI} \Longleftarrow i[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$ (Rule (interI) on 1)

Case. $\dfrac{\begin{array}{c}\Sigma, \vec{x}{:}\vec{\sigma}, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \le x_1, x_2 \le u_2; E; \Pi, \pi : s_1 \circ [x_1, x_2] \\ \vdash^{\nu} V \Longleftarrow s_2 \circ [x_1, x_2] \searrow \mathcal{C}; \mathcal{I}\end{array}}{\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} \texttt{impI}\ (x_1.x_2.\pi.V) \Longleftarrow s_1 \supset s_2 \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}}\supset\text{I}$

To show: $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{impI}\ (x_1.x_2.\pi.(V[\vec{t_0}/\vec{x}])) \Longleftarrow s_1[\vec{t_0}/\vec{x}] \supset s_2[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$

1. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi[\vec{t_0}/\vec{x}], u_1[\vec{t_0}/\vec{x}] \leq x_1, x_2 \leq u_2[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}], \pi : s_1[\vec{t_0}/\vec{x}] \circ [x_1, x_2] \vdash^{\nu[\vec{t_0}/\vec{x}]} V[\vec{t_0}/\vec{x}] \Longleftarrow s_2[\vec{t_0}/\vec{x}] \circ [x_1, x_2]$ (i.h. on premise)
2. $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0, E[\vec{t_0}/\vec{x}]; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \texttt{impI}\ (x_1.x_2.\pi.(V[\vec{t_0}/\vec{x}])) \Longleftarrow s_1[\vec{t_0}/\vec{x}] \supset s_2[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$ (Rule ($\supset$I) on 1)

□

Lemma C.4 (Completeness)**.** *Let $\vec{x}$ be a list of term variables, $\vec{\sigma}$ a list of sorts, and $\vec{t_0}$ a list of terms such that:*

1. $\Sigma \vdash \vec{t_0} : \vec{\sigma}$
2. *Variables in $\vec{x}$ do not appear in the formulas of Π (they may appear in top level judgment annotations like k $\mathsf{claims}\ \cdot$ and $\cdot \circ [u_1, u_2]$).*
3. *Variables in $\vec{x}$ do not appear in E_0.*

Then,

A. *If variables from $\vec{x}$ do not appear in s and V and $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V \Longleftarrow s \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$, then for any E, there are $\mathcal{C}$ and $\mathcal{I}$ such that:*
   (a) $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$.
   (b) *For every $(\Sigma', \vec{x}{:}\vec{\sigma}; \Psi' \models c') \in \mathcal{C}$, it is the case that $\Sigma'; \Psi'[\vec{t_0}/\vec{x}] \models c'[\vec{t_0}/\vec{x}]$.*
   (c) *For every $(\Sigma', \vec{x}{:}\vec{\sigma}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E_0, E' \models i'$.*

B. *If variables from $\vec{x}$ do not appear in R and $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R \Longrightarrow s[\vec{t_0}/\vec{x}] \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$, then variables from $\vec{x}$ do not appear in s and for any E, there are $\mathcal{C}$ and $\mathcal{I}$ such that:*
   (a) $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$.
   (b) *For every $(\Sigma', \vec{x}{:}\vec{\sigma}; \Psi' \models c') \in \mathcal{C}$, it is the case that $\Sigma'; \Psi'[\vec{t_0}/\vec{x}] \models c'[\vec{t_0}/\vec{x}]$.*
   (c) *For every $(\Sigma', \vec{x}{:}\vec{\sigma}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E_0, E' \models i'$.*

Proof. By simultaneous induction on given derivations of $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V \Longleftarrow s \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$ and $\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R \Longrightarrow s \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]$, and case analysis of their last rules. The proof is tedious but straightforward. Some representative cases are shown below. (Note that the required conditions (a), (b), and (c) are identical for A and B.)

Case.
$$\frac{\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R \Longrightarrow s \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]] \quad \Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_1[\vec{t_0}/\vec{x}] \leq u_1'[\vec{t_0}/\vec{x}] \quad \Sigma; \Psi[\vec{t_0}/\vec{x}] \models u_2'[\vec{t_0}/\vec{x}] \leq u_2[\vec{t_0}/\vec{x}]}{\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} R \Longleftarrow s \circ [u_1'[\vec{t_0}/\vec{x}], u_2'[\vec{t_0}/\vec{x}]]}\text{infer}$$

1. There exist $\mathcal{C}'$ and $\mathcal{I}'$ such that $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}'; \mathcal{I}'$

(i.h. on premise; by condition in A, $s = s[\vec{t_0}/\vec{x}]$)

2. (b) holds for $\mathcal{C}'$ (i.h.)
3. (c) holds for $\mathcal{I}'$ (i.h.)
4. Let $\mathcal{C}_1 = (\Sigma; \Psi \models u_1 \leq u_1'), (\Sigma; \Psi \models u_2' \leq u_2)$. Choose $\mathcal{C} = \mathcal{C}', \mathtt{unsat}(\mathcal{C}_1)$ and $\mathcal{I} = \mathcal{I}'$.
5. $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} R \Longleftarrow s \circ [u_1', u_2'] \searrow \mathcal{C}; \mathcal{I}$ (Rule (infer) on 1)

(a) is the same as 5 above. (b) holds for $\mathcal{C}$ because of 2, and because of the 2nd and 3rd premises. (c) holds for $\mathcal{I}$ because of 3.

Case. $$\frac{\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} V \Longleftarrow s \circ [u_1, u_2]}{\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \mathtt{check}\ V\ s\ u_1\ u_2 \Longrightarrow s[\vec{t_0}/\vec{x}] \circ [u_1, u_2]}\text{check}$$

1. Variables in $\vec{x}$ do not appear in $\mathtt{check}\ V\ s\ u_1\ u_2$ (Condition in B)
2. Variables in $\vec{x}$ do not appear in V and s (From 1)
3. There are $\mathcal{C}'$ and $\mathcal{I}'$ such that $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2] \searrow \mathcal{C}'; \mathcal{I}'$

(i.h. on premise; can be used due to 2)

4. $\mathcal{C}'$ satisfies (b) (i.h.)
5. $\mathcal{I}'$ satisfies (c) (i.h.)
6. Choose $\mathcal{C} = \mathcal{C}'$ and $\mathcal{I} = \mathcal{I}'$
7. $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} \mathtt{check}\ V\ s\ u_1\ u_2 \Longrightarrow s \circ [u_1, u_2] \searrow \mathcal{C}'; \mathcal{I}'$

(a) is the same as 7. (b) and (c) hold because of 4 and 5 repsectively.

Case. $$\frac{\Sigma; E_0 \models i}{\Sigma; \Psi[\vec{t_0}/\vec{x}]; E_0; \Pi[\vec{t_0}/\vec{x}] \vdash^{\nu[\vec{t_0}/\vec{x}]} \mathtt{interI} \Longleftarrow i \circ [u_1[\vec{t_0}/\vec{x}], u_2[\vec{t_0}/\vec{x}]]}\text{interI}$$

1. Choose $\mathcal{C} = \cdot$ and $\mathcal{I} = \mathtt{unsat}(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i)$
2. $\Sigma, \vec{x}{:}\vec{\sigma}; \Psi; E; \Pi \vdash^{\nu} \mathtt{interI} \Longleftarrow i \circ [u_1, u_2] \searrow \mathcal{C}; \mathcal{I}$ (Rule (interI))

(a) is the same as 2. (b) holds vacuously since $\mathcal{C} = \cdot$. To prove (c) we consider two cases.

- $(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i) \notin \mathcal{I}$: (c) is vacuously true since $\mathcal{I}$ must be empty.
- $(\Sigma, \vec{x}{:}\vec{\sigma}; E \models i) \in \mathcal{I}$: We must show that $\Sigma; E, E_0 \models i$. This follows by applying (S-weaken) from §4.2.1 to the premise.

□

Theorem C.5 (Completeness of PCFS verification; Theorem 5.12)**.** *Suppose that* $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u, u]$*. Let* ctime *be a fresh constant. Then there exist* $\mathcal{C}$ *and* $\mathcal{I}$ *such that the following hold.*

1. $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow s \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$.
2. *For each* $(\Sigma', \mathsf{ctime{:}time}; \Psi' \models c') \in \mathcal{C}$*, it is the case that* $\Sigma'; \Psi'[u/\mathsf{ctime}] \models c'[u/\mathsf{ctime}]$.
3. *For each* $(\Sigma', \mathsf{ctime{:}time}; E' \models i') \in \mathcal{I}$*, it is the case that* $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$.

Proof. We proceed as follows.

1. $\Sigma; \cdot; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u, u]$ (Assumption)
2. $\Sigma; \cdot; E; \Pi[u/\mathsf{ctime}] \vdash^{\nu[u/\mathsf{ctime}]} V \Longleftarrow s \circ [\mathsf{ctime}[u/\mathsf{ctime}], \mathsf{ctime}[u/\mathsf{ctime}]]$ (ctime is fresh)
3. There are $\mathcal{C}$ and $\mathcal{I}$ such that $\Sigma, \mathsf{ctime{:}time}; \cdot; \cdot; \Pi \vdash^{\nu} V \Longleftarrow s \circ [\mathsf{ctime}, \mathsf{ctime}] \searrow \mathcal{C}; \mathcal{I}$ (Lemma C.4 on 2)
4. (2) holds (Also by Lemma C.4)
5. For each $(\Sigma', \mathsf{ctime{:}time}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E, E' \models i'$ (Also by Lemma C.4)
6. For each $(\Sigma', \mathsf{ctime{:}time}; E' \models i') \in \mathcal{I}$, it is the case that $\Sigma'; E, E'[u/\mathsf{ctime}] \models i'[u/\mathsf{ctime}]$ (5; E', i' cannot contain ctime)

□

C.3 Proofs from §5.3

Theorem C.6 (Correctness; Theorem 5.19)**.** *If* $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$*, then* $\Sigma; \Psi; E; \Pi \vdash^{\nu} V \Longleftarrow s \circ [u_1, u_2]$.

Proof. By induction on the given derivation of $\Sigma; \Psi; E; \Pi \xrightarrow{\nu} V : s \circ [u_1, u_2]$ and case analysis of its last rule. Some representative cases are shown below.

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Pi, \pi : k \ \mathsf{claims}\ s \circ [u_1, u_2], \tau : s \circ [u_1, u_2] \xrightarrow{\nu} V : r \circ [u_1', u_2'] \\ \nu = k', u_b, u_e \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \qquad \Sigma; \Psi \models k \succeq k'\end{array}}{\Sigma; \Psi; E; \Pi, \pi : k \ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} V[\pi/\tau] : r \circ [u_1', u_2']}\mathsf{claims}$

To show: $\Sigma; \Psi; E; \Pi, \pi : k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} V[\pi/\tau] \Longleftarrow r \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Pi, \pi : k \ \mathsf{claims}\ s \circ [u_1, u_2], \tau : s \circ [u_1, u_2] \vdash^{\nu} V \Longleftarrow r \circ [u_1', u_2']$ (i.h. on 1st premise)
2. $\Sigma; \Psi; E; \Pi, \pi : k \ \mathsf{claims}\ s \circ [u_1, u_2] \vdash^{\nu} \pi \Longrightarrow s \circ [u_1, u_2]$

(Rule (claims) on premises 3–5)

3. $\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{claims}\; s \circ [u_1, u_2] \vdash^{\nu} V[\pi/\tau] \Longleftarrow r \circ [u_1', u_2']$ (Theorem 5.7 on 2,3)

Case. $\dfrac{\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2], \tau : k \;\mathsf{claims}\; s \circ [u_1, u_2] \xrightarrow{\nu} V : r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2] \xrightarrow{\nu} \mathtt{saysE}\ \pi\ (\tau.V) : r \circ [u_1', u_2']}\mathsf{saysL}$

To show: $\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2] \vdash^{\nu} \mathtt{saysE}\ \pi\ (\tau.V) \Longleftarrow r \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2], \tau : k \;\mathsf{claims}\; s \circ [u_1, u_2] \vdash^{\nu} V \Longleftarrow r \circ [u_1', u_2']$

(i.h. on premise)

2. $\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2] \vdash^{\nu} \pi \Longrightarrow k \;\mathsf{says}\; s \circ [u_1, u_2]$ (Rule (hyp))

3. $\Sigma; \Psi; E; \Pi, \pi : k \;\mathsf{says}\; s \circ [u_1, u_2] \vdash^{\nu} \mathtt{saysE}\ \pi\ (\tau.V) \Longleftarrow r \circ [u_1', u_2']$

(Rule (saysE) on 1,2)

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} V_1 : s_1 \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2], \tau : s_2 \circ [u_1', u_2'] \xrightarrow{\nu} V_2 : r \circ [u_1'', u_2''] \\ \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} V_2[(\mathtt{impE}\ \pi\ V_1\ u_1'\ u_2')/\tau] : r \circ [u_1'', u_2'']}\supset\mathsf{L}$

To show: $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} V_2[(\mathtt{impE}\ \pi\ V_1\ u_1'\ u_2')/\tau] : r \circ [u_1'', u_2'']$

1. $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} V_1 \Longleftarrow s_1 \circ [u_1', u_2']$ (i.h. on 1st premise)

2. $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} \pi \Longrightarrow s_1 \supset s_2 \circ [u_1, u_2]$ (Rule (hyp))

3. $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \vdash^{\nu} \mathtt{impE}\ \pi\ V_1\ u_1'\ u_2' \Longrightarrow s_2 \circ [u_1', u_2']$

(Rule ($\supset$E) on 2,1 and 3rd,4th premises)

4. $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2], \tau : s_2 \circ [u_1', u_2'] \vdash^{\nu} V_2 \Longleftarrow r \circ [u_1'', u_2'']$

(i.h. on 2nd premise)

5. $\Sigma; \Psi; E; \Pi, \pi : s_1 \supset s_2 \circ [u_1, u_2] \xrightarrow{\nu} V_2[(\mathtt{impE}\ \pi\ V_1\ u_1'\ u_2')/\tau] : r \circ [u_1'', u_2'']$

(Theorem 5.7 on 3,4)

□

Appendix D

Proofs from §6

D.1 Soundness of Goal-directed Search

Lemma D.1. *If* $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}$ *and* $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} \mathcal{Q}$*, then* $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$.

Proof. By induction on the derivation of $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}$ and case analysis of its last rule. The interesting cases are shown below. Note that the cases (F-$\supset_1$) and (F-@) do not apply since the boolean in their conclusions is always $\mathtt{tt}$.

Case. $\dfrac{}{\Sigma; p \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (u_1 \leq u_1') :: (u_2' \leq u_2) :: []; \mathtt{ff}}$ F-init

To show: $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$.

1. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} (u_1 \leq u_1') :: (u_2' \leq u_2) :: []$ (Assumption)
2. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} (u_2' \leq u_2) :: []$ (Inversion on 1)
3. $\Sigma; \Psi \models u_2' \leq u_2$ (Inversion on 2)

The required conclusions are contained in 2 and 3.

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ [u_1', u_2']) :: \mathcal{Q}; \mathtt{ff}}$ F-$\supset_2$

To show: $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$.

1. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} (g_1 \circ [u_1', u_2']) :: \mathcal{Q}$ (Assumption)
2. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} \mathcal{Q}$ (Inversion on 1)
3. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$ (i.h. on premise and 2)

□

Lemma D.2. *Suppose the following hold.*

1. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$

Then,

A. $b = \mathtt{ff}$ *implies* $\Sigma; \Psi; E; \Gamma, d \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$

B. $b = \mathtt{tt}$ *implies* $\Sigma; \Psi; E; \Gamma, d \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ *for every* u_b *and* u_e.

Proof. We prove (A) and (B) by a simultaneous induction on the derivation given in (1).

Proof of (A)

We case analyze the last rule in the derivation given in (1).

Case. $\dfrac{}{\Sigma; p \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (u_1 \leq u_1') :: (u_2' \leq u_2) :: []; \mathtt{ff}}$ F-init

To show: $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi \models u_1' \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2'$ ((C-refl-time) from §4.2.1)
2. $\Sigma; \Psi; E; \Gamma, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule (init) on 1)

Case. $\dfrac{\Sigma; d_1 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}$ F-$\wedge_1$

To show: $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma, d_1 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (A) on premise and assumption 2)
2. $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_1', u_2'], d_1 \circ [u_1', u_2'], d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Weakening Theorem 4.8(1d) on 1)
3. $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\wedge$L) on 2)

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}$ F-$\wedge_2$

Similar to the previous case.

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ [u_1', u_2']) :: \mathcal{Q}; \mathtt{ff}}$ F-$\supset_2$

To show: $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} (g_1 \circ [u_1', u_2']) :: \mathcal{Q}$ (Assumption 2)

2. The following hold (Inversion on 1)
   (a) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} g_1 \circ [u_1', u_2']$
   (b) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$
3. $\Sigma; \Psi; E; \Gamma, d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (A) on premise and 2b)
4. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_1', u_2'] \xrightarrow{\nu} g_1 \circ [u_1', u_2']$ (Weakening Theorem 4.8(1d) on 2a)
5. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_1', u_2'], d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$
   (Weakening Theorem 4.8(1d) on 3)
6. $\Sigma; \Psi \models u_1' \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2'$ ((C-refl-time) from §4.2.1)
7. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\supset$L) on 4–6)

Case. $\dfrac{\Sigma \vdash t : \sigma \qquad \Sigma; d[t/x] \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; \forall x{:}\sigma.d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}$F-$\forall$

To show: $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma, d[t/x] \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (A) on premise and assumption 2)
2. $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_1', u_2'], d[t/x] \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$
   (Weakening Theorem 4.8(1d) on 1)
3. $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\forall$L) on 2 and 1st premise)

Proof of (B)

We case analyze the last rule in the derivation given in (1).

Case. $\dfrac{\Sigma; d_1 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}$F-$\wedge_1$

To show: $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma, d_1 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (B) on premise and assumption 2)
2. $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_b, u_e], d_1 \circ [u_b, u_e], d_2 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$
   (Weakening Theorem 4.8(1d) on 1)
3. $\Sigma; \Psi; E; \Gamma, d_1 \wedge d_2 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\wedge$L) on 2)

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}{\Sigma; d_1 \wedge d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}\text{F-}\wedge_2$

Similar to the previous case.

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ \phi) :: \mathcal{Q}; \mathtt{tt}}\text{F-}\supset_1$

To show: $\Sigma; \Psi; E; \Gamma; g_1 \supset d_2 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} (g_1 \circ \phi) :: \mathcal{Q}$ (Assumption 2)
2. The following hold (Inversion on 1)
   (a) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} g_1 \circ \phi$
   (b) $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$
3. $\Sigma; \Psi; E; \Gamma, d_2 \circ \phi \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (B) on premise and 2b, choosing $[u_b, u_e] = \phi$)
4. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_b, u_e] \xrightarrow{\nu} g_1 \circ \phi$ (Weakening Theorem 4.8(1d) on 2a)
5. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_b, u_e], d_2 \circ \phi \xrightarrow{\nu} p \circ [u_1', u_2']$ (Weakening Theorem 4.8(1d) on 3)
6. $\Sigma; \Psi \models u_b \leq \infty$ and $\Sigma; \Psi \models -\infty \leq u_e$ ((C-refl-time) from §4.2.1)
7. $\Sigma; \Psi; E; \Gamma, g_1 \supset d_2 \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\supset$L) on 4–6; $\phi = [+\infty, -\infty]$)

Case. $\dfrac{\Sigma \vdash t : \sigma \qquad \Sigma; d[t/x] \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}{\Sigma; \forall x{:}\sigma.d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}\text{F-}\forall$

To show: $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Gamma, d[t/x] \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (i.h. (B) on premise and assumption 2)
2. $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_b, u_e], d[t/x] \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$
   (Weakening Theorem 4.8(1d) on 1)
3. $\Sigma; \Psi; E; \Gamma, \forall x{:}\sigma.d \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule ($\forall$L) on 2 and 1st premise)

Case. $\dfrac{\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b}{\Sigma; d \mathbin{@} [u_1'', u_2''] \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{tt}}\text{F-@}$

To show: $\Sigma; \Psi; E; \Gamma, d \mathbin{@} [u_1'', u_2''] \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$

1. Case analysis of b.
   - Case: $b = \mathtt{tt}$
     (a) $\Sigma; \Psi; E; \Gamma, d \circ [u_1'', u_2''] \xrightarrow{\nu} p \circ [u_1', u_2']$

(i.h. (B) on premise and assumption 2 choosing $[u_b, u_e] = [u_1'', u_2'']$)

- Case: $b = \texttt{ff}$
  (a) $\Sigma; \Psi; E; \Gamma, d \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$
  (i.h. (A) on premise and assumption 2)
  (b) $\Sigma; \Psi \models u_1'' \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2''$
  (Lemma D.1 on premise and assumption 2)
  (c) $\Sigma; \Psi; E; \Gamma, d \circ [u_1'', u_2''] \xrightarrow{\nu} d \circ [u_1', u_2']$ (Identity Theorem 4.13 using b)
  (d) $\Sigma; \Psi; E; \Gamma, d \circ [u_1'', u_2''] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Cut Theorem 4.12 on c,a)

$\Sigma; \Psi; E; \Gamma, d \circ [u_1'', u_2''] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Result of case analysis)

2. $\Sigma; \Psi; E; \Gamma, d \mathbin{@} [u_1'', u_2''] \circ [u_b, u_e], d \circ [u_1'', u_2''] \xrightarrow{\nu} p \circ [u_1', u_2']$
(Weakening Theorem 4.8(1d) on 1)

3. $\Sigma; \Psi; E; \Gamma, d \mathbin{@} [u_1'', u_2''] \circ [u_b, u_e] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule (@L) on 2)

□

Lemma D.3 (Soundness of F-sequents; Lemma 6.1)**.** *Suppose the following hold.*

1. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \mathcal{Q}$

Then, $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$.

Proof. We case analyze b.

Case. $b = \texttt{tt}$.

1. $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$
(Lemma D.2(B) on assumptions 1,2 choosing $[u_b, u_e] = [u_1, u_2]$)

Case. $b = \texttt{ff}$.

1. $\Sigma; \Psi; E; \Gamma, d \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Lemma D.2(A) on assumptions 1,2)
2. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$ (Lemma D.1 on assumptions 1,2)
3. $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \xrightarrow{\nu} d \circ [u_1', u_2']$ (Identity Theorem 4.13 on 2)
4. $\Sigma; \Psi; E; \Gamma, d \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Cut Theorem 4.12 on 1,3)

□

Theorem D.4 (Soundness; Theorem 6.2)**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, |\Xi| \overset{\nu}{\rightharpoonup} g \circ [u_1, u_2]$

C. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} p \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} \mathcal{Q}$

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

Case. $\dfrac{\Sigma; \Psi; E; \Delta| \overset{k,u_1,u_2}{\Rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} k \text{ says } g \circ [u_1, u_2]}$R-says

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} k \text{ says } g \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta| \xrightarrow{k,u_1,u_2} g \circ [u_1, u_2]$ (i.h. on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} k \text{ says } g \circ [u_1, u_2]$ (Rule (saysR) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Delta, k \text{ claims } d \circ [u_1, u_2]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta; \Xi :: (k \text{ says } d \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']}$L-says

To show: $\Sigma; \Psi; E; \Delta, |\Xi|, k \text{ says } d \circ [u_1, u_2] \overset{\nu}{\rightharpoonup} g \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Delta, k \text{ claims } d \circ [u_1, u_2], |\Xi| \overset{\nu}{\rightharpoonup} g \circ [u_1', u_2']$ (i.h. on premise)
2. $\Sigma; \Psi; E; \Delta, k \text{ claims } d \circ [u_1, u_2], |\Xi|, k \text{ says } d \circ [u_1, u_2] \overset{\nu}{\rightharpoonup} g \circ [u_1', u_2']$
   (Weakening Theorem 4.8(1d) on 1)
3. $\Sigma; \Psi; E; \Delta, |\Xi|, k \text{ says } d \circ [u_1, u_2] \overset{\nu}{\rightharpoonup} g \circ [u_1', u_2']$ (Rule (saysL) on 2)

Case. $\dfrac{d \circ [u_1, u_2] \in \Delta \qquad \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}$N-clause

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} \mathcal{Q}$ (i.h. on 3rd premise)
2. $\Sigma; \Psi; E; \Delta, d \circ [u_1, u_2] \overset{\nu}{\rightharpoonup} p \circ [u_1', u_2']$ (Lemma D.3 on 2nd premise and 1)
3. $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} p \circ [u_1', u_2']$ (Contraction Theorem 4.8(2) on 2 using 1st premise)

Case. $\dfrac{\begin{array}{c} k \text{ claims } d \circ [u_1, u_2] \in \Delta \qquad \nu = k_0, u_b, u_e \\ \Sigma; \Psi \models k \succeq k_0 \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \\ \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}$N-claims

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightharpoonup} p \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} \mathcal{Q}$ (i.h. on 7th premise)
2. $\Sigma; \Psi; E; \Delta, d \circ [u_1, u_2] \xrightarrow{\nu} p \circ [u_1', u_2']$ (Lemma D.3 on 6th premise and 1)
3. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} p \circ [u_1', u_2']$ (Rule (claims) on 2 and 1st–5th premises)

□

D.2 Properties of Goal-directed Search

Lemma D.5 (Weakening)**.** *The following hold.*

1. *If* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *then*
   (a) $\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$
   (b) $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$
   (c) $\Sigma; \Psi; E, i; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$
   (d) $\Sigma; \Psi; E; \Delta, J \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$
2. *If* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *then*
   (a) $\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$
   (b) $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$
   (c) $\Sigma; \Psi; E, i; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$
   (d) $\Sigma; \Psi; E; \Delta, J \overset{\nu}{\Rightarrow} \mathcal{Q}$
3. *If* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *then,*
   (a) $\Sigma, x{:}\sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$
   (b) $\Sigma; \Psi, c; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$
   (c) $\Sigma; \Psi; E, i; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$
   (d) $\Sigma; \Psi; E; \Delta, J; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$
4. *If* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *then*
   (a) $\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$
   (b) $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$
   (c) $\Sigma; \Psi; E, i; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$
   (d) $\Sigma; \Psi; E; \Delta, J \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$
5. *If* $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ *then* $\Sigma, x{:}\sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$

Further, all constructed derivations have depths less than or equal to those of given derivations.

Proof. (5) follows by induction on the given derivation. (1a), (2a), (3a), and (4a), then follow by a simultaneous induction on given derivations. Similarly, (1b)–(4b), (1c)–(4c), and (1d)–(4d) follow by separate simultaneous inductions. □

Lemma D.6 (Constraint substitution)**.** *Suppose* $\Sigma; \Psi \models c$*. Then the following hold.*

1. $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$
2. $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$
3. $\Sigma; \Psi, c; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$
4. $\Sigma; \Psi, c; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$

Proof. By simultaneous induction on given derivations and case analysis of their last rules. For the cases of rules (R-cons), (Q-leq), and (N-claims) we appeal to assumption (C-cut) from §4.2.1, as in the proof of Lemma B.3. □

Lemma D.7 (View subsumption)**.** *Suppose the following hold:*

1. $\nu = k_0, u_b, u_e$
2. $\Sigma; \Psi \models k_0 \succeq k_0'$*,* $\Sigma; \Psi \models u_b \leq u_b'$*, and* $\Sigma; \Psi \models u_e' \leq u_e$*.*
3. $\nu' = k_0', u_b', u_e'$

Then,

A. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu'}{\Rightarrow} g \circ [u_1, u_2]$ *by a derivation of less or equal depth.*

B. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu'}{\Rightarrow} \mathcal{Q}$ *by a derivation of less or equal depth.*

C. $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu'}{\Leftarrow} g \circ [u_1, u_2]$ *by a derivation of less or equal depth.*

D. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu'}{\Leftrightarrow} p \circ [u_1, u_2]$ *by a derivation of less or equal depth.*

Proof. By simultaneous induction on derivations given in (A)–(D) and case analysis of their last rules. The only interesting case is (N-claims), where we appeal to assumptions (C-trans-time) and (C-trans-prin) from §4.2.1. □

Lemma D.8 (Time subsumption)**.** *Suppose* $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$*. Then the following hold.*

1. *If* $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1, u_2] \searrow \mathcal{Q}; b$ *and* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ *then there is a* $\mathcal{Q}'$ *such that* $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}'; b$ *and* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}'$*.*

2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1', u_2']$
3. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$
4. $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1', u_2']$

Proof. (1) follows by an induction on the given derivation of $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1, u_2] \searrow \mathcal{Q}; b$. Proof of (2) is shown below. (3) and (4) then follow by a simultaneous induction on the depths of the given derivations and case analysis of last rules, using (2) for the case of rule (R-N). Further Lemma D.6 is needed for the case of rule (R-$\supset$) and Lemma D.7 is needed for the case of (R-says), as in the proof of Theorem 4.11. (Note that the induction must be on the depth of derivations, not on the structure of derivations, because in the case of rule (R-says), we appeal to the i.h. after using Lemma D.7.)

Proof of (2). We case analyze the rule used to derive $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1, u_2]$.

Case. $\dfrac{d \circ [u_1'', u_2''] \in \Delta \qquad \Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1, u_2] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1, u_2]}$ N-clause

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1', u_2']$

1. There exists $\mathcal{Q}'$ such that
   (a) $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}'; b$
   (b) $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}'$ (Clause (1) of theorem on 2nd and 3rd premises)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1', u_2']$ (Rule (N-clause) on 1st premise, 1a, 1b)

Case. $\dfrac{\begin{array}{c} k \text{ claims } d \circ [u_1'', u_2''] \in \Delta \qquad \nu = k_0, u_b, u_e \\ \Sigma; \Psi \models k \succeq k_0 \qquad \Sigma; \Psi \models u_1'' \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2'' \\ \Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1, u_2] \searrow \mathcal{Q}; b \qquad \Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1, u_2]}$ N-claims

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1', u_2']$

1. There exists $\mathcal{Q}'$ such that
   (a) $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}'; b$
   (b) $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}'$ (Clause (1) of theorem on 6th and 7th premises)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Leftarrow} p \circ [u_1', u_2']$ (Rule (N-claims) on 1st–5th premises, 1a, 1b)

□

Lemma D.9 (Left subsumption for F-sequents)**.** *Suppose the following hold:*

1. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$

3. $\Sigma; \Psi \models u_1'' \leq u_1$ *and* $\Sigma; \Psi \models u_2 \leq u_2''$

Then, there is a $\mathcal{Q}'$ *such that*

A. $\Sigma; d \circ [u_1'', u_2''] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}'; b$

B. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}'$

Proof. By induction on the given derivation of $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ and case analysis of its last rule. The only interesting case is shown below.

Case. $\dfrac{}{\Sigma; p \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (u_1 \leq u_1') :: (u_2' \leq u_2) :: []; \mathtt{ff}}$F-init

To show: There is $\mathcal{Q}'$ such that (A) and (B) hold (with $d = p$). We claim that $\mathcal{Q}' = (u_1'' \leq u_1') :: (u_2' \leq u_2'') :: []$ satisfies these properties. (A) follows immediately from rule (F-init). (B) is proved as follows.

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (u_1 \leq u_1') :: (u_2' \leq u_2) :: []$ (Assumption 2)
2. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$ (Inversion on 1)
3. $\Sigma; \Psi \models u_1'' \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2''$

   ((C-trans-time) from §4.2.1 on 2 and assumption 3)
4. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (u_1'' \leq u_1') :: (u_2' \leq u_2'') :: []$ (Rules (Q-[]) and (Q-leq) on 3)

□

Lemma D.10. *If* $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}$ *and* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$*, then* $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$*.*

Proof. By induction on the derivation of $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}$ and case analysis of its last rule. The interesting cases are shown below. Note that the cases (F-$\supset_1$) and (F-@) do not apply since the boolean in their conclusions is always $\mathtt{tt}$.

Case. $\dfrac{}{\Sigma; p \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (u_1 \leq u_1') :: (u_2' \leq u_2) :: []; \mathtt{ff}}$F-init

To show: $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$.

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (u_1 \leq u_1') :: (u_2' \leq u_2) :: []$ (Assumption)
2. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (u_2' \leq u_2) :: []$ (Inversion on 1)
3. $\Sigma; \Psi \models u_2' \leq u_2$ (Inversion on 2)

The required conclusions are contained in 2 and 3.

Case. $\dfrac{\Sigma; d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; \mathtt{ff}}{\Sigma; g_1 \supset d_2 \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow (g_1 \circ [u_1', u_2']) :: \mathcal{Q}; \mathtt{ff}}$F-$\supset_2$

To show: $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$.

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} (g_1 \circ [u_1', u_2']) :: \mathcal{Q}$ (Assumption)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \mathcal{Q}$ (Inversion on 1)
3. $\Sigma; \Psi \models u_1 \leq u_1'$ and $\Sigma; \Psi \models u_2' \leq u_2$ (i.h. on premise and 2)

□

Lemma D.11 (Admissibility of ($\supset$L))**.** *Suppose the following hold:*

1. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$
2. $\Sigma; \Psi \models u_1 \leq u_1'$ *and* $\Sigma; \Psi \models u_2' \leq u_2$

Then,

A. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} g'' \circ [u_1'', u_2'']$ *implies* $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g'' \circ [u_1'', u_2'']$

B. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}$

C. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2']; \Xi \overset{\nu}{\Leftarrow} g'' \circ [u_1'', u_2'']$ *implies* $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2]; \Xi \overset{\nu}{\Leftarrow} g'' \circ [u_1'', u_2'']$

D. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Leftrightarrow} p'' \circ [u_1'', u_2'']$ *implies* $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p'' \circ [u_1'', u_2'']$

Proof. By simultaneous induction on the depths of the derivations given in (A)–(D) and case analysis of the last rules in them. The cases in (A), (B), and (C) are straightforward – we apply the induction hypothesis to the premises of the last rule and reapply the rule. In order to apply the induction hypothesis to the premises of the rules (L-clause), (L-cons), (L-inter), and (L-$\exists$) in the proof of (C), we appeal to Lemma D.5 to weaken the given derivation of $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ appropriately. The cases in the proof of (D) are shown below.

Case. $\dfrac{\begin{array}{c} d' \circ [u_3, u_4] \in (\Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2']) \\ \Sigma; d' \circ [u_3, u_4] \ll p \circ [u_3', u_4'] \searrow \mathcal{Q}; b \\ \Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Leftrightarrow} p \circ [u_3', u_4']}$N-clause

To show: $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u_3', u_4']$. We consider three subcases on the 1st premise and b:

Subcase. $d' \circ [u_3, u_4] \in (\Delta, g \supset d \circ [u_1, u_2])$

1. $d' \circ [u_3, u_4] \in (\Delta, g \supset d \circ [u_1, u_2])$ (Subcase assumption)
2. $\Sigma; d' \circ [u_3, u_4] \ll p \circ [u'_3, u'_4]$ (2nd premise)
3. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
4. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u'_3, u'_4]$ (Rule (N-clause) on 1–3)

Subcase. $d' \circ [u_3, u_4] = d \circ [u'_1, u'_2]$ and $b = \mathtt{tt}$. So, $d' = d$, $u_3 = u'_1$, and $u_4 = u'_2$.

1. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
2. $\Sigma; d \circ [u'_1, u'_2] \ll p \circ [u'_3, u'_4] \searrow \mathcal{Q}; \mathtt{tt}$ (2nd premise and subcase assumption)
3. $\Sigma; \Psi \models u_1 \leq u'_1$ and $\Sigma; \Psi \models u'_2 \leq u_2$ (Assumption 2)
4. There is a $\mathcal{Q}'$ such that
   (a) $\Sigma; d \circ [u_1, u_2] \ll p \circ [u'_3, u'_4] \searrow \mathcal{Q}'; \mathtt{tt}$
   (b) $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}'$ (Lemma D.9 on 1–3)
5. $\Sigma; g \supset d \circ [u_1, u_2] \ll p \circ [u'_3, u'_4] \searrow (g \circ \phi) :: \mathcal{Q}'; \mathtt{tt}$ (Rule (F-$\supset_1$) on 4a)
6. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ \phi$ (Lemma D.8 on assumption 1)
7. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} (g \circ \phi) :: \mathcal{Q}'$ (Rule (Q-goal) on 4b and 6)
8. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u'_3, u'_4]$ (Rule (N-clause) on 5,7)

Subcase. $d' \circ [u_3, u_4] = d \circ [u'_1, u'_2]$ and $b = \mathtt{ff}$. So, $d' = d$, $u_3 = u'_1$, and $u_4 = u'_2$.

1. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
2. $\Sigma; d \circ [u'_1, u'_2] \ll p \circ [u'_3, u'_4] \searrow \mathcal{Q}; \mathtt{ff}$ (2nd premise and subcase assumption)
3. $\Sigma; \Psi \models u_1 \leq u'_1$ and $\Sigma; \Psi \models u'_2 \leq u_2$ (Assumption 2)
4. There is a $\mathcal{Q}'$ such that
   (a) $\Sigma; d \circ [u_1, u_2] \ll p \circ [u'_3, u'_4] \searrow \mathcal{Q}'; \mathtt{ff}$
   (b) $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}'$ (Lemma D.9 on 1–3)
5. $\Sigma; g \supset d \circ [u_1, u_2] \ll p \circ [u'_3, u'_4] \searrow (g \circ [u'_3, u'_4]) :: \mathcal{Q}'; \mathtt{ff}$ (Rule (F-$\supset_2$) on 4a)
6. $\Sigma; \Psi \models u'_1 \leq u'_3$ and $\Sigma; \Psi \models u'_4 \leq u'_2$ (Lemma D.10 on 1,2)
7. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u'_3, u'_4]$ (Lemma D.8 on 6 and assumption 1)
8. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} (g \circ [u'_3, u'_4]) :: \mathcal{Q}'$ (Rule (Q-goal) on 4b and 7)
9. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u'_3, u'_4]$ (Rule (N-clause) on 5,8)

Case. $$\frac{\begin{array}{c} k \;\mathsf{claims}\; d' \circ [u_3, u_4] \in (\Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2']) \qquad \nu = k_0, u_b, u_e \\ \Sigma; \Psi \models k \succeq k_0 \qquad \Sigma; \Psi \models u_3 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_4 \\ \Sigma; d' \circ [u_3, u_4] \ll p \circ [u_3', u_4'] \searrow \mathcal{Q}; b \\ \Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Leftrightarrow} p \circ [u_3', u_4']}\text{N-claims}$$

To show: $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u_3', u_4']$

1. $k \;\mathsf{claims}\; d' \circ [u_3, u_4] \in \Delta$ (Follows from 1st premise)
2. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 7th premise)
3. $\Sigma; \Psi; E; \Delta, g \supset d \circ [u_1, u_2] \overset{\nu}{\Leftrightarrow} p \circ [u_3', u_4']$

(Rule (N-claims) on 1, 2nd–6th premises, and 2)

□

Lemma D.12 (Admissibility of (claims))**.** *Suppose the following hold:*

1. $\nu = k_0, u_b, u_e$
2. $\Sigma; \Psi \models k_0' \succeq k_0$
3. $\Sigma; \Psi \models u_b' \leq u_b$
4. $\Sigma; \Psi \models u_e \leq u_e'$

Then,

A. $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q}$

C. $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e']; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e']; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$

Proof. By simultaneous induction on the depths of derivations given in (A)–(D) and case analysis of their last rules. The interesting cases are shown below.

Case. $$\frac{\Sigma; \Psi; E; (\Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'])| \overset{k, u_1, u_2}{\Rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} k \;\mathsf{says}\; g \circ [u_1, u_2]}\text{R-says}$$

To show: $\Sigma; \Psi; E; \Delta, k_0' \;\mathsf{claims}\; d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} k \;\mathsf{says}\; g \circ [u_1, u_2]$

1. $(\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'])| = (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'])|$ (Definition)
2. $\Sigma; \Psi; E; (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'])| \overset{k, u_1, u_2}{\Rightarrow} g \circ [u_1, u_2]$ (Premise and 1)
3. $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} k \text{ says } g \circ [u_1, u_2]$ (Rule (R-says) on 2)

Case. $\dfrac{\begin{array}{c} d \circ [u_1, u_2] \in (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e']) \\ \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \\ \Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}$ N-clause

To show: $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']$. We consider two subcases on the 1st premise.

Subcase. $d \circ [u_1, u_2] \in (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'])$

1. $d \circ [u_1, u_2] \in (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'])$ (Subcase assumption)
2. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ (2nd premise)
3. $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
4. $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']$ (Rule (N-clause) on 1–3)

Subcase. $d \circ [u_1, u_2] = d_0' \circ [u_b', u_e']$. Therefore, $d = d_0'$, $u_1 = u_b'$, and $u_2 = u_e'$.

1. $k_0' \text{ claims } d_0' \circ [u_b', u_e'] \in (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'])$
2. $\nu = k_0, u_b, u_e$ (Assumption 1)
3. $\Sigma; \Psi \models k_0' \succeq k_0$ (Assumption 2)
4. $\Sigma; \Psi \models u_b' \leq u_b$ (Assumption 3)
5. $\Sigma; \Psi \models u_e \leq u_e'$ (Assumption 4)
6. $\Sigma; d_0' \circ [u_b', u_e'] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b$ (2nd premise and subcase assumption)
7. $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
8. $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']$ (Rule (N-claims) on 1–7)

Case. $\dfrac{\begin{array}{c} k \text{ claims } d \circ [u_1, u_2] \in (\Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e']) \qquad \nu = k_0, u_b, u_e \\ \Sigma; \Psi \models k \succeq k_0 \qquad \Sigma; \Psi \models u_1 \leq u_b \qquad \Sigma; \Psi \models u_e \leq u_2 \\ \Sigma; d \circ [u_1, u_2] \ll p \circ [u_1', u_2'] \searrow \mathcal{Q}; b \\ \Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'], d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']}$ N-claims

To show: $\Sigma; \Psi; E; \Delta, k_0' \text{ claims } d_0' \circ [u_b', u_e'] \overset{\nu}{\Leftrightarrow} p \circ [u_1', u_2']$

1. k claims $d \circ [u_1, u_2] \in (\Delta, k'_0$ claims $d'_0 \circ [u'_b, u'_e])$ (Follows from 1st premise)
2. $\Sigma; \Psi; E; \Delta, k'_0$ claims $d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 7th premise)
3. $\Sigma; \Psi; E; \Delta, k'_0$ claims $d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u'_1, u'_2]$

(Rule (N-claims) on 1, 2nd–6th premises, and 2)

□

Lemma D.13 (Admissibility of (∧L))**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$

C. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$

Proof. By simultaneous induction on depths of derivations given in (A)–(D) and case analysis of their last rules. One interesting case is shown below.

Case.
$$\dfrac{\begin{array}{c} d \circ [u_1, u_2] \in (\Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e]) \\ \Sigma; d \circ [u_1, u_2] \ll p \circ [u'_1, u'_2] \searrow \mathcal{Q}; b \\ \Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q} \end{array}}{\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e], d_0 \circ [u'_b, u'_e], d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u'_1, u'_2]}\text{N-clause}$$

To show: $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u'_1, u'_2]$. We analyze three subcases on the 1st premise.

Subcase. $d \circ [u_1, u_2] \in (\Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e])$

1. $d \circ [u_1, u_2] \in (\Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e])$ (Subcase assumption)
2. $\Sigma; d \circ [u_1, u_2] \ll p \circ [u'_1, u'_2] \searrow \mathcal{Q}; b$ (2nd premise)
3. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)
4. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u'_1, u'_2]$ (Rule (N-clause) on 1–3)

Subcase. $d \circ [u_1, u_2] = d_0 \circ [u'_b, u'_e]$. Then, $d = d_0$, $u_1 = u'_b$, and $u_2 = u'_e$.

1. $d_0 \wedge d'_0 \circ [u'_b, u'_e] \in (\Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e])$

2. $\Sigma; d_0 \circ [u'_b, u'_e] \ll p \circ [u'_1, u'_2] \searrow \mathcal{Q}; b$ (2nd premise and subcase assumption)

3. $\Sigma; d_0 \wedge d'_0 \circ [u'_b, u'_e] \ll p \circ [u'_1, u'_2] \searrow \mathcal{Q}; b$ (Rule (F-$\wedge_1$) on 2)

4. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ (i.h. on 3rd premise)

5. $\Sigma; \Psi; E; \Delta, d_0 \wedge d'_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u'_1, u'_2]$ (Rule (N-clause) on 1,3,4)

Subcase. $d \circ [u_1, u_2] = d'_0 \circ [u'_b, u'_e]$. This subcase is similar to the previous subcase, except that we use rule (F-$\wedge_2$) in the third step. □

Lemma D.14 (Admissibility of (∀L))**.** *Suppose* $\Sigma \vdash t : \sigma$*. Then the following hold.*

A. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e], d_0[t/x] \circ [u_b, u_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e], d_0[t/x] \circ [u_b, u_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$

C. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e], d_0[t/x] \circ [u_b, u_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e], d_0[t/x] \circ [u_b, u_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d_0 \circ [u_b, u_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$

Proof. By simultaneous induction on depths of derivations given in (A)–(D) and case analysis of their last rules, as in the proof of Lemma D.13. In the case of rule (N-clause), we use rule (F-∀). □

Lemma D.15 (Admissibility of (@L))**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e], d_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e] \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$

B. $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e], d_0 \circ [u'_b, u'_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e] \overset{\nu}{\Rightarrow} \mathcal{Q}$

C. $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e], d_0 \circ [u'_b, u'_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e]; \Xi \overset{\nu}{\Leftarrow} g \circ [u_1, u_2]$

D. $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e], d_0 \circ [u'_b, u'_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta, d_0 \mathbin{@} [u'_b, u'_e] \circ [u_b, u_e] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$

Proof. By simultaneous induction on depths of derivations given in (A)–(D) and case analysis of their last rules, as in the proof of Lemma D.13. In the case of rule (N-clause), we use rule (F-@). □

D.3 Properties of the Sequent Calculus

Lemma D.16 (Strong right inversion for $\wedge, \supset, \forall, @$)**.** *The following hold in the sequent calculus for BL.*

1. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \wedge s_2 \circ [u_1, u_2]$ *implies both* $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \circ [u_1, u_2]$ *and* $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_2 \circ [u_1, u_2]$ *by derivations of smaller or equal depth.*
2. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]$ *implies* $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ *by a derivation of smaller or equal depth.*
3. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} \forall x{:}\sigma.s \circ [u_1, u_2]$ *implies* $\Sigma, x{:}\sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1, u_2]$ *by a derivation of smaller or equal depth.*
4. $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*

Proof. Each statement follows by a separate induction on the depth of the given derivation and a case analysis of the last rule in the derivation. For every statement, only one right rule may end the derivation, in which case the result follows from the premise(s) of the rule. For left rules, we apply the induction hypothesis to relevant premises and reapply the rule. As an illustration, we show some representative cases in the proof of statement (2).

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \supset\mathrm{R}$

To show: $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ by a derivation of smaller or equal depth. This follows immediately, since it is the premise of the rule. Further, the derivation ending at the premise has a depth one less than that of the whole derivation. Note also that the given derivation in statement (2) cannot end in any other right rule.

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], r_1 \circ [u_1', u_2'] \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2] \\ \Sigma; \Psi; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], r_2 \circ [u_1', u_2'] \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]\end{array}}{\Sigma; \Psi; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'] \xrightarrow{\nu} s_1 \supset s_2 \circ [u_1, u_2]} \vee\mathrm{L}$

To show: $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ by a derivation of smaller or equal depth. Let the depth of the entire given derivation be n. So each premise has depth at most $n - 1$.

1. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], r_1 \circ [u_1', u_2'], s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ by a derivation of depth at most $n - 1$ (i.h. on 1st premise)
2. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], r_2 \circ [u_1', u_2'], s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ by a derivation of depth at most $n - 1$ (i.h. on 2nd premise)
3. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Gamma, r_1 \vee r_2 \circ [u_1', u_2'], s_1 \circ [x_1, x_2] \xrightarrow{\nu} s_2 \circ [x_1, x_2]$ by a derivation of depth at most n (Rule ($\vee$L) on 1,2)

□

Lemma D.17 (Limited strong right inversion for $c, i, \vee, \exists, \mathsf{says}$)**.** *The following hold in the sequent calculus for BL. (Observe that the hypotheses in the following statements are restricted to the form Δ.)*

1. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} c \circ [u_1, u_2]$ *implies* $\Sigma; \Psi \models c$
2. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} i \circ [u_1, u_2]$ *implies* $\Sigma; E \models i$
3. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} s_1 \vee s_2 \circ [u_1, u_2]$ *implies either* $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} s_1 \circ [u_1, u_2]$ *or* $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} s_2 \circ [u_1, u_2]$*, in each case by a derivation of strictly smaller depth.*
4. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} \exists x{:}\sigma.s \circ [u_1, u_2]$ *implies that there is a t such that $\Sigma \vdash t : \sigma$ and $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} s[t/x] \circ [u_1, u_2]$ by a derivation of strictly smaller depth.*
5. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]$ *implies* $\Sigma; \Psi; E; \Delta| \xrightarrow{k,u_1,u_2} s \circ [u_1, u_2]$ *by a derivation of strictly smaller depth.*

Proof. Each statement follows by a separate induction on the depth of the given derivation and a case analysis of the last rule in the derivation, as in the proof of Lemma D.16. As an illustration, we show some representative cases in the proof of (5).

Case. $\dfrac{\Sigma; \Psi; E; \Delta| \xrightarrow{k,u_1,u_2} s \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]}\mathsf{saysR}$

To show: $\Sigma; \Psi; E; \Delta| \xrightarrow{k,u_1,u_2} s \circ [u_1, u_2]$ by a derivation of strictly smaller depth. This follows immediately from the premise. Note also that no other right rules apply.

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Delta, s_1 \supset s_2 \circ [u_1'', u_2''] \xrightarrow{\nu} s_1 \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Delta, s_1 \supset s_2 \circ [u_1'', u_2''], s_2 \circ [u_1', u_2'] \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2] \\ \Sigma; \Psi \models u_1'' \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2''\end{array}}{\Sigma; \Psi; E; \Delta, s_1 \supset s_2 \circ [u_1'', u_2''] \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]}\supset\mathrm{L}$

To show: $\Sigma; \Psi; E; (\Delta, s_1 \supset s_2 \circ [u_1'', u_2''])| \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]$ by a derivation of strictly smaller depth. Let the depth of the entire derivation be n. Then the depth of each premise is at most $n - 1$.

1. $\Sigma; \Psi; E; (\Delta, s_1 \supset s_2 \circ [u_1'', u_2''], s_1 \circ [u_1', u_2'])| \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]$ by a derivation of depth at most $n - 2$ (i.h. on 2nd premise)
2. $(\Delta, s_1 \supset s_2 \circ [u_1'', u_2''], s_1 \circ [u_1', u_2'])| = \Delta| = (\Delta, s_1 \supset s_2 \circ [u_1'', u_2''])|$ (Definition)
3. $\Sigma; \Psi; E; (\Delta, s_1 \supset s_2 \circ [u_1'', u_2''])| \xrightarrow{\nu} k\ \mathsf{says}\ s \circ [u_1, u_2]$ by a derivation of depth at most $n - 2$ (1,2)

Due to the syntax of Δ, the left rules ($\vee$L), (saysL), (consL), (interL), ($\bot$L), and ($\exists$L) do not apply in the proof of any of the statements of the theorem. As the reader may easily

check, the induction step for statement (5) would not have succeeded for the cases (saysL), (consL), (interL), and ($\bot$L) had they been relevant. Similarly, the induction step for cases ($\vee$L) and ($\exists$L) would not succeed for statements (3) and (4) respectively. □

Lemma D.18 (Strong left inversion for $c, i, \wedge, \vee, \top, \exists, \mathsf{says}, @$; Lemma 6.4)**.** *The following hold for the sequent calculus of BL.*

1. $\Sigma; \Psi; E; \Gamma, c \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi, c; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
2. $\Sigma; \Psi; E; \Gamma, i \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E, i; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
3. $\Sigma; \Psi; E; \Gamma, s_1 \wedge s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2], s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
4. $\Sigma; \Psi; E; \Gamma, s_1 \vee s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies both* $\Sigma; \Psi; E; \Gamma, s_1 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *and* $\Sigma; \Psi; E; \Gamma, s_2 \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *by derivations of smaller or equal depth.*
5. $\Sigma; \Psi; E; \Gamma, \top \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
6. $\Sigma; \Psi; E; \Gamma, \exists x{:}\sigma.s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma, x{:}\sigma; \Psi; E; \Gamma, s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
7. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*
8. $\Sigma; \Psi; E; \Gamma, s\ @\ [u_1'', u_2''] \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ *implies* $\Sigma; \Psi; E; \Gamma, s \circ [u_1'', u_2''] \xrightarrow{\nu} r \circ [u_1', u_2']$ *by a derivation of smaller or equal depth.*

Proof. Each statement follows by a separate induction on the depth of the given derivation and a case analysis of the last rule in the derivation. As an illustration, we show some representative cases in the proof of statement (7).

Case. $\dfrac{\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2], k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, k\ \mathsf{says}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\mathsf{saysL}$ (principal case)

To show: $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of smaller or equal depth. Let the depth of the entire derivation be n. Then the premise has a derivation of depth $n - 1$.

1. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2], k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of depth less than or equal to $n - 1$ (i.h. on the premise)
2. $\Sigma; \Psi; E; \Gamma, k\ \mathsf{claims}\ s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of depth less than or equal to $n - 1$ (Contraction Theorem 4.8(2) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Gamma, k'' \text{ says } s'' \circ [u_1'', u_2''], k'' \text{ claims } s'' \circ [u_1'', u_2''], \quad k \text{ says } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, k'' \text{ says } s'' \circ [u_1'', u_2''], k \text{ says } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']}\text{saysL (other case)}$

To show: $\Sigma; \Psi; E; \Gamma, k'' \text{ says } s'' \circ [u_1'', u_2''], k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of shorter or equal depth. Let the depth of the entire derivation be n. Then the premise has a derivation of depth $n - 1$.

1. $\Sigma; \Psi; E; \Gamma, k'' \text{ says } s'' \circ [u_1'', u_2''], k'' \text{ claims } s'' \circ [u_1'', u_2''], k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of depth at most $n - 1$ (i.h. on premise)

2. $\Sigma; \Psi; E; \Gamma, k'' \text{ says } s'' \circ [u_1'', u_2''], k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} r \circ [u_1', u_2']$ by a derivation of depth at most n (Rule (saysL) on 1)

Case. $\dfrac{\Sigma; \Psi; E; (\Gamma, k \text{ says } s \circ [u_1, u_2])| \xrightarrow{k', u_1', u_2'} s' \circ [u_1', u_2']}{\Sigma; \Psi; E; \Gamma, k \text{ says } s \circ [u_1, u_2] \xrightarrow{\nu} k' \text{ says } s' \circ [u_1', u_2']}\text{saysR}$

To show: $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} k' \text{ says } s' \circ [u_1', u_2']$ by a derivation of smaller or equal depth. Let the depth of the entire derivation be n. Then the premise has a derivation of depth $n - 1$.

1. $(\Gamma, k \text{ says } s \circ [u_1, u_2])| = \Gamma|$ (Definition)

2. $\Sigma; \Psi; E; \Gamma| \xrightarrow{k', u_1', u_2'} s' \circ [u_1', u_2']$ by a derivation of depth $n - 1$ (Premise and 1)

3. $\Sigma; \Psi; E; \Gamma|, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{k', u_1', u_2'} s' \circ [u_1', u_2']$ by a derivation of depth at most $n - 1$ (Weakening Theorem 4.8(1d) on 2)

4. $\Gamma|, k \text{ claims } s \circ [u_1, u_2] = (\Gamma, k \text{ claims } s \circ [u_1, u_2])|$ (Definition)

5. $\Sigma; \Psi; E; (\Gamma, k \text{ claims } s \circ [u_1, u_2])| \xrightarrow{k', u_1', u_2'} s' \circ [u_1', u_2']$ by a derivation of depth at most $n - 1$ (3,4)

6. $\Sigma; \Psi; E; \Gamma, k \text{ claims } s \circ [u_1, u_2] \xrightarrow{\nu} k' \text{ says } s' \circ [u_1', u_2']$ by a derivation of depth at most n (Rule (saysR) on 5)

□

D.4 Completeness of Goal-directed Search

We define the size of chunks h and groups Ξ as follows.

$$\begin{array}{rcc}
\mathtt{size}(d) & = & 1 \\
\mathtt{size}(c) & = & 1 \\
\mathtt{size}(i) & = & 1 \\
\mathtt{size}(h_1 \wedge h_2) & = & 1 + \mathtt{size}(h_1) + \mathtt{size}(h_2) \\
\mathtt{size}(h_1 \vee h_2) & = & 1 + \mathtt{size}(h_1) + \mathtt{size}(h_2) \\
\mathtt{size}(\top) & = & 1 \\
\mathtt{size}(\bot) & = & 1 \\
\mathtt{size}(\exists x{:}\sigma.h) & = & 1 + \mathtt{size}(h) \\
\mathtt{size}(k \text{ says } d) & = & 1 \\
\mathtt{size}(h \mathbin{@} [u_1, u_2]) & = & 1 + \mathtt{size}(h) \\
 & & \\
\mathtt{size}([]) & = & 0 \\
\mathtt{size}(\Xi :: (h \circ [u_1, u_2])) & = & \mathtt{size}(\Xi) + \mathtt{size}(h)
\end{array}$$

Theorem D.19 (Completeness; Theorem 6.5)**.** *The following hold.*

A. $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ *implies* $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_0 \circ [u_0, u_0']$

B. $\Sigma; \Psi; E; \Delta, |\Xi| \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ *implies* $\Sigma; \Psi; E; \Delta; \Xi \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$

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

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

Proof of (A)

To prove (A), we case analyze the last rule in the given derivation of $\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_0 \circ [u_0, u_0']$. For right rules we apply the i.h. to premises and then apply the corresponding rule from R-sequents. For left rules as well as the rule (claims) we apply the i.h. to the premises, and use one of Lemmas D.11, D.12, D.13, D.14, and D.15, depending on the principal connective.

Case. $\dfrac{\Sigma;\Psi \models u_1' \le u_1 \qquad \Sigma;\Psi \models u_2 \le u_2'}{\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \xrightarrow{\nu} p \circ [u_1, u_2]}\text{init}$

To show: $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} p \circ [u_1, u_2]$

1. $\Sigma; p \circ [u_1', u_2'] \ll p \circ [u_1, u_2] \searrow (u_1' \le u_1) :: (u_2 \le u_2') :: []; \mathtt{ff}$ (Rule (F-init))
2. $\Sigma;\Psi \models u_1' \le u_1$ (1st premise)
3. $\Sigma;\Psi \models u_2 \le u_2'$ (2nd premise)
4. $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} []$ (Rule (Q-[]))
5. $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} (u_2 \le u_2') :: []$ (Rule (Q-leq) on 3,4)
6. $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} (u_1' \le u_1) :: (u_2 \le u_2') :: []$ (Rule (Q-leq) on 2,5)
7. $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Leftrightarrow} p \circ [u_1, u_2]$ (Rule (N-clause) on 1,6)
8. $\Sigma;\Psi;E;\Delta, p \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} p \circ [u_1, u_2]$ (Rule (R-N) on 7)

Case. $\dfrac{\begin{array}{c}\Sigma;\Psi;E;\Delta, k\ \mathsf{claims}\ d \circ [u_1, u_2], d \circ [u_1, u_2] \xrightarrow{\nu} g \circ [u_1', u_2'] \\ \nu = k', u_b, u_e \qquad \Sigma;\Psi \models u_1 \le u_b \qquad \Sigma;\Psi \models u_e \le u_2 \qquad \Sigma;\Psi \models k \succeq k'\end{array}}{\Sigma;\Psi;E;\Delta, k\ \mathsf{claims}\ d \circ [u_1, u_2] \xrightarrow{\nu} g \circ [u_1', u_2']}\text{claims}$

To show: $\Sigma;\Psi;E;\Delta, k\ \mathsf{claims}\ d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$

1. $\Sigma;\Psi;E;\Delta, k\ \mathsf{claims}\ d \circ [u_1, u_2], d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ (i.h. (A) on premise)
2. $\Sigma;\Psi;E;\Delta, k\ \mathsf{claims}\ d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$

(Lemma D.12(A) on 1 and 2nd–5th premises)

Case. $\dfrac{\Sigma;\Psi;E;\Delta| \xrightarrow{k,u_1,u_2} g \circ [u_1, u_2]}{\Sigma;\Psi;E;\Delta \xrightarrow{\nu} k\ \mathsf{says}\ g \circ [u_1, u_2]}\mathsf{saysR}$

To show: $\Sigma;\Psi;E;\Delta \overset{\nu}{\Rightarrow} k\ \mathsf{says}\ g \circ [u_1, u_2]$

1. $\Sigma;\Psi;E;\Delta| \overset{k,u_1,u_2}{\Rightarrow} g \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma;\Psi;E;\Delta \overset{\nu}{\Rightarrow} k\ \mathsf{says}\ g \circ [u_1, u_2]$ (Rule (R-says) on 1)

Case. $\dfrac{\Sigma;\Psi;E;\Delta \xrightarrow{\nu} g \circ [u_1, u_2]}{\Sigma;\Psi;E;\Delta \xrightarrow{\nu} g\ @\ [u_1, u_2] \circ [u_1', u_2']}@\text{R}$

To show: $\Sigma;\Psi;E;\Delta \overset{\nu}{\Rightarrow} g\ @\ [u_1, u_2] \circ [u_1', u_2']$

1. $\Sigma;\Psi;E;\Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma;\Psi;E;\Delta \overset{\nu}{\Rightarrow} g\ @\ [u_1, u_2] \circ [u_1', u_2']$ (Rule (R-@) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Delta, d \mathbin{@} [u_1', u_2'] \circ [u_1, u_2], d \circ [u_1', u_2'] \xrightarrow{\nu} g \circ [u_1'', u_2'']}{\Sigma; \Psi; E; \Delta, d \mathbin{@} [u_1', u_2'] \circ [u_1, u_2] \xrightarrow{\nu} g \circ [u_1'', u_2'']}$@L

To show: $\Sigma; \Psi; E; \Delta, d \mathbin{@} [u_1', u_2'] \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$

1. $\Sigma; \Psi; E; \Delta, d \mathbin{@} [u_1', u_2'] \circ [u_1, u_2], d \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta, d \mathbin{@} [u_1', u_2'] \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$ (Lemma D.15(A) on 1)

Case. $\dfrac{\Sigma; \Psi \models c}{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} c \circ [u_1, u_2]}$consR

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} c \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} c \circ [u_1, u_2]$ (Rule (R-cons) on premise)

Case. $\dfrac{\Sigma; E \models i}{\Sigma; \Psi; E; \Gamma \xrightarrow{\nu} i \circ [u_1, u_2]}$interR

To show: $\Sigma; \Psi; E; \Gamma \overset{\nu}{\Rightarrow} i \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Gamma \overset{\nu}{\Rightarrow} i \circ [u_1, u_2]$ (Rule (R-inter) on premise)

Case. $\dfrac{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_1 \circ [u_1, u_2] \qquad \Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_1 \wedge g_2 \circ [u_1, u_2]}$∧R

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \wedge g_2 \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \circ [u_1, u_2]$ (i.h. (A) on 1st premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_2 \circ [u_1, u_2]$ (i.h. (A) on 2nd premise)
3. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \wedge g_2 \circ [u_1, u_2]$ (Rule (R-∧) on 1,2)

Case. $\dfrac{\Sigma; \Psi; E; \Delta, d_1 \wedge d_2 \circ [u_1, u_2], d_1 \circ [u_1, u_2], d_2 \circ [u_1, u_2] \xrightarrow{\nu} g \circ [u_1', u_2']}{\Sigma; \Psi; E; \Delta, d_1 \wedge d_2 \circ [u_1, u_2] \xrightarrow{\nu} g \circ [u_1', u_2']}$∧L

To show: $\Sigma; \Psi; E; \Delta, d_1 \wedge d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Delta, d_1 \wedge d_2 \circ [u_1, u_2], d_1 \circ [u_1, u_2], d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta, d_1 \wedge d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ (Lemma D.13(A) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_1 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \xrightarrow{\nu} g_1 \vee g_2 \circ [u_1, u_2]}$∨R$_1$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]$ (Rule (R-$\vee_1$) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g_2 \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g_1 \vee g_2 \circ [u_1, u_2]} \vee \mathrm{R}_2$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_2 \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_1 \vee g_2 \circ [u_1, u_2]$ (Rule (R-$\vee_2$) on 1)

Case. $\dfrac{}{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} \top \circ [u_1, u_2]} \top \mathrm{R}$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \top \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \top \circ [u_1, u_2]$ (Rule (R-$\top$))

Case. $\dfrac{\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Delta, h \circ [x_1, x_2] \overset{\nu}{\rightarrow} g \circ [x_1, x_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} h \supset g \circ [u_1, u_2]} \supset \mathrm{R}$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} h \supset g \circ [u_1, u_2]$

1. $\Sigma, x_1{:}\mathsf{time}, x_2{:}\mathsf{time}; \Psi, u_1 \leq x_1, x_2 \leq u_2; E; \Delta; h \circ [x_1, x_2] \overset{\nu}{\Leftarrow} g \circ [x_1, x_2]$ (i.h. (B) on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} h \supset g \circ [u_1, u_2]$ (Rule (R-$\supset$) on 1)

Case. $\dfrac{\begin{array}{c}\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_1 \circ [u_1', u_2'] \\ \Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2], d_2 \circ [u_1', u_2'] \overset{\nu}{\rightarrow} g \circ [u_1'', u_2''] \\ \Sigma; \Psi \models u_1 \leq u_1' \qquad \Sigma; \Psi \models u_2' \leq u_2\end{array}}{\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2] \overset{\nu}{\rightarrow} g \circ [u_1'', u_2'']} \supset \mathrm{L}$

To show: $\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$

1. $\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g_1 \circ [u_1', u_2']$ (i.h. (A) on 1st premise)
2. $\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2], d_2 \circ [u_1', u_2'] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$ (i.h. (A) on 2nd premise)
3. $\Sigma; \Psi; E; \Delta, g_1 \supset d_2 \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1'', u_2'']$ (Lemma D.11(A) on 1,2 and 3rd,4th premises)

Case. $\dfrac{\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g \circ [u_1, u_2]}{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} \forall x{:}\sigma. g \circ [u_1, u_2]} \forall \mathrm{R}$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \forall x{:}\sigma. g \circ [u_1, u_2]$

1. $\Sigma, x{:}\sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \forall x{:}\sigma.g \circ [u_1, u_2]$ (Rule (R-$\forall$) on 1)

Case. $\dfrac{\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d \circ [u_1, u_2], d[t/x] \circ [u_1, u_2] \overset{\nu}{\rightarrow} g \circ [u_1', u_2'] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d \circ [u_1, u_2] \overset{\nu}{\rightarrow} g \circ [u_1', u_2']} \forall\text{L}$

To show: $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$

1. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d \circ [u_1, u_2], d[t/x] \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta, \forall x{:}\sigma.d \circ [u_1, u_2] \overset{\nu}{\Rightarrow} g \circ [u_1', u_2']$ (Lemma D.14(A) on 1 and 2nd premise)

Case. $\dfrac{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g[t/x] \circ [u_1, u_2] \qquad \Sigma \vdash t : \sigma}{\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} \exists x{:}\sigma.g \circ [u_1, u_2]} \exists\text{R}$

To show: $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \exists x{:}\sigma.g \circ [u_1, u_2]$

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g[t/x] \circ [u_1, u_2]$ (i.h. (A) on premise)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} \exists x{:}\sigma.g \circ [u_1, u_2]$ (Rule (R-$\exists$) on 1 and 2nd premise)

Due to syntactic restrictions on Δ, no other rules apply.

Proof of (B)

To prove (B), we subinduct on $\texttt{size}(\Xi)$. If $\texttt{size}(\Xi) = 0$, then $\Xi = []$. In this case we proceed as follows.

1. $\Sigma; \Psi; E; \Delta \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta \overset{\nu}{\Rightarrow} g_0 \circ [u_0, u_0']$ (i.h. (A) on 1; valid because (B) > (A))
3. $\Sigma; \Psi; E; \Delta; [] \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-R) on 2)

If, on the other hand, $\texttt{size}(\Xi) > 0$, then there is at least one chunk in Ξ. We case analyze the form of the last chunk in Ξ.

Case. $\Xi = \Xi' :: (d \circ [u_1, u_2])$

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, d \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, d \circ [u_1, u_2]; \Xi' \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (i.h. (B) on 1; $\texttt{size}(\Xi') < \texttt{size}(\Xi)$)
3. $\Sigma; \Psi; E; \Delta; \Xi' :: (d \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-clause) on 2)

Case. $\Xi = \Xi' :: (c \circ [u_1, u_2])$

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, c \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi, c; E; \Delta, |\Xi'| \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Lemma D.18(1) on 1)
3. $\Sigma; \Psi, c; E; \Delta; \Xi' \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (c \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (Rule (L-cons) on 3)

Case. $\Xi = \Xi' :: (i \circ [u_1, u_2])$

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, i \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E, i; \Delta, |\Xi'| \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Lemma D.18(2) on 1)
3. $\Sigma; \Psi; E, i; \Delta; \Xi' \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (i \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (Rule (L-inter) on 3)

Case. $\Xi = \Xi' :: (h_1 \wedge h_2 \circ [u_1, u_2])$. Define $\Xi'' = \Xi' :: (h_1 \circ [u_1, u_2]) :: (h_2 \circ [u_1, u_2])$. Note that $\mathtt{size}(\Xi'') = \mathtt{size}(\Xi) - 1$.

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, h_1 \wedge h_2 \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, |\Xi'|, h_1 \circ [u_1, u_2], h_2 \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Lemma D.18(3) on 1)
3. $\Sigma; \Psi; E; \Delta; \Xi' :: (h_1 \circ [u_1, u_2]) :: (h_2 \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi'') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (h_1 \wedge h_2 \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (Rule (L-$\wedge$) on 3)

Case. $\Xi = \Xi' :: (h_1 \vee h_2 \circ [u_1, u_2])$. Define $\Xi_1 = \Xi' :: (h_1 \circ [u_1, u_2])$ and $\Xi_2 = \Xi' :: (h_2 \circ [u_1, u_2])$. Note that $\mathtt{size}(\Xi_1) < \mathtt{size}(\Xi)$ and $\mathtt{size}(\Xi_2) < \mathtt{size}(\Xi)$.

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, h_1 \vee h_2 \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, |\Xi'|, h_1 \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ and $\Sigma; \Psi; E; \Delta, |\Xi'|, h_2 \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Lemma D.18(4) on 1)
3. $\Sigma; \Psi; E; \Delta; \Xi' :: (h_1 \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ and $\Sigma; \Psi; E; \Delta; \Xi' :: (h_2 \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi_1) < \mathtt{size}(\Xi)$ and $\mathtt{size}(\Xi_2) < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (h_1 \vee h_2 \circ [u_1, u_2]) \xLeftarrow{\nu} g_0 \circ [u_0, u_0']$ (Rule (L-$\vee$) on derivations in 3)

Case. $\Xi = \Xi' :: (\top \circ [u_1, u_2])$. Note that $\mathtt{size}(\Xi') < \mathtt{size}(\Xi)$.

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, \top \circ [u_1, u_2] \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, |\Xi'| \xrightarrow{\nu} g_0 \circ [u_0, u_0']$ (Lemma D.18(5) on 1)

3. $\Sigma; \Psi; E; \Delta; \Xi' \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (\top \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-$\top$) on 3)

Case. $\Xi = \Xi' :: (\bot \circ [u_1, u_2])$

1. $\Sigma; \Psi; E; \Delta; \Xi' :: (\bot \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-$\bot$))

Case. $\Xi = \Xi' :: (\exists x{:}\sigma.h \circ [u_1, u_2])$. Define $\Xi'' = \Xi', h \circ [u_1, u_2]$ and note that $\mathtt{size}(\Xi'') < \mathtt{size}(\Xi)$.

1. $\Sigma; \Psi; E; \Delta, |\Xi'|, \exists x{:}\sigma.h \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma, x{:}\sigma; \Psi; E; \Delta, |\Xi'|, h \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Lemma D.18(6) on 1)
3. $\Sigma, x{:}\sigma; \Psi; E; \Delta; \Xi' :: (h \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi'') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (\exists x{:}\sigma.h \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-$\exists$) on 3)

Case. $\Xi = \Xi' :: (k \text{ says } d \circ [u_1, u_2])$.

1. $\Sigma; \Psi; E; \Delta, \Xi', k \text{ says } d \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, \Xi', k \text{ claims } d \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Lemma D.18(7) on 1)
3. $\Sigma; \Psi; E; \Delta, k \text{ claims } d \circ [u_1, u_2]; \Xi' \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (k \text{ says } d \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-says) on 3)

Case. $\Xi = \Xi' :: (h \mathbin{@} [u_1', u_2'] \circ [u_1, u_2])$. Define $\Xi'' = \Xi' :: (h \circ [u_1', u_2'])$ and note that $\mathtt{size}(\Xi'') < \mathtt{size}(\Xi)$.

1. $\Sigma; \Psi; E; \Delta, \Xi', h \mathbin{@} [u_1', u_2'] \circ [u_1, u_2] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Given derivation)
2. $\Sigma; \Psi; E; \Delta, \Xi', h \circ [u_1', u_2'] \overset{\nu}{\rightarrow} g_0 \circ [u_0, u_0']$ (Lemma D.18(8) on 1)
3. $\Sigma; \Psi; E; \Delta; \Xi' :: (h \circ [u_1', u_2']) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (i.h. (B) on 2; $\mathtt{size}(\Xi'') < \mathtt{size}(\Xi)$)
4. $\Sigma; \Psi; E; \Delta; \Xi' :: (h \mathbin{@} [u_1', u_2'] \circ [u_1, u_2]) \overset{\nu}{\Leftarrow} g_0 \circ [u_0, u_0']$ (Rule (L-@) on 3)

□

Bibliography

[1] FUSE: Filesystem in Userspace. Available from http://fuse.sourceforge.net/.

[2] OpenSSL: The open source toolkit for SSL/TLS. Online at http://www.openssl.org.

[3] SecPAL research release for .NET, 2007. Available from http://research.microsoft.com/en-us/projects/secpal/.

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

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

[6] Martín Abadi. Variations in access control logic. In *Ninth International Conference on Deontic Logic in Computer Science (DEON 2008)*, pages 96–109, 2008.

[7] Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *Conference Record of the 26th Sympoisum on Principles Of Programming Languages (POPL'99)*, pages 147–160. ACM Press, January 1999.

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

[9] V. Michele Abrusci. Additional results on intuitionistic linear propositional logic. Technical Report 6, Department of Philosophy, University of Bari, Italy, October 1988.

[10] Marcos K. Aguilera, Minwen Ji, Mark Lillibridge, John MacCormick, Erwin Oertli, Dave Andersen, Mike Burrows, Timothy Mann, and Chandramohan A. Thekkath. Block-level security for network-attached disks. In *Proceedings of the 2nd Conference on File and Storage Technologies (FAST)*, pages 159–174, 2003.

[11] Natasha Alechina, Michael Mendler, Valeria de Paiva, and Eike Ritter. Categorical and Kripke semantics for constructive S4 modal logic. In *CSL '01: Proceedings of the 15th International Workshop on Computer Science Logic*, pages 292–307, 2001.

[12] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

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

[14] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise Privacy Authorization Language (EPAL 1.2), 2003. Online at http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/index.html.

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

[16] Henk Barendregt and Silvia Ghilezan. Lambda terms for natural deduction, sequent calculus and cut elimination. *Journal of Functional Programming*, 10(1):121–134, 2000.

[17] Adam Barth and John C. Mitchell. Managing digital rights using linear logic. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 127–136, 2006.

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

[19] Lujo Bauer, Lorrie Cranor, Robert W. Reeder, Michael K. Reiter, and Kami Vaniea. A user study of policy creation in a flexible access-control system. In *CHI 2008: Conference on Human Factors in Computing Systems*, pages 543–552, April 2008.

[20] 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 '05)*, pages 431–445, September 2005.

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

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

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

[24] Moritz Y. Becker, Jason F. Mackay, and Blair Dillaway. Abductive authorization credential gathering. In *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 1–8, 2009.

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

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

[27] P. N. Benton, Gavin M. Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications (TLCA '93)*, pages 75–90, 1993.

[28] P.N. Benton, G.M. Bierman, and V.C.V. de Paiva. Computational types from a logical perspective. *Journal of Functional Programming*, 8(2):177–193, 1998.

[29] Gavin Bierman and Valeria de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65:383–416, 2000.

[30] P. Blackburn, J. van Benthem, and F. Wolter. *Handbook of Modal Logic*. Elsevier B. V., 2007.

[31] M. Blaze, J. Fiegenbaum, and J. Ioannidis. The Keynote trust-management system version 2. See http://www.ietf.org/rfc/rfc2704.txt, 1999.

[32] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.

[33] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Washington, DC, USA, 1996. IEEE Computer Society.

[34] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael K. Reiter. Consumable credentials in logic-based access-control systems. In *Electronic Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07)*, 2007. Online at http://www.isoc.org/isoc/conferences/ndss/07/papers/consumable_credentials.pdf.

[35] Torben Braüner and Valeria de Paiva. Towards constructive hybrid logic. In *Electronic Proceedings of Methods for Modalities 3 (M4M3)*, 2003. Online at http://m4m.loria.fr/M4M3/Papers/brauner.ps.gz.

[36] Glenn Bruns and Michael Huth. Access-control policies via Belnap logic: Effective and efficient composition and analysis. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF-21)*, pages 163–176, 2008.

[37] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. Audit-based compliance control. *International Journal of Information Security*, 6(2):133–151, 2007.

[38] Iliano Cervesato. Proof-theoretic foundation of compilation in logic programming languages. In *Proceedings of the 1998 Joint International Conference and Symposium on Logic Programming (JICSLP'98)*, pages 115–129, 1998.

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

[40] Avik Chaudhuri. On secure distributed implementations of dynamic access control. In *Proceedings of the Joint Workshop on Foundations of Computer Security, Automated Reasoning for Security Protocol Analysis, and Issues in the Theory of Security (FCS-ARSPA-WITS)*, pages 93–107, 2008.

[41] Avik Chaudhuri and Deepak Garg. PCAL: Language support for proof-carrying authorization systems. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, pages 184–199, 2009.

[42] Avik Chaudhuri, Prasad Naldurg, Sriram K. Rajamani, G. Ramalingam, and Lakshmisubrahmanyam Velaga. Eon: modeling and analyzing dynamic access control systems with logic programs. In *Proceedings of the 15th ACM conference on Computer and Communications Security (CCS'08)*, pages 381–390, 2008.

[43] Kaustuv Chaudhuri. *The Focused Inverse Method for Linear Logic*. PhD thesis, Carnegie Mellon University, December 2006.

[44] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[45] Andrew Cirillo, Radha Jagadeesan, Corin Pitcher, and James Riely. Do As I SaY! Programmatic access control with explicit identities. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF-20)*, pages 16–30, 2007.

[46] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[47] Russell Coker. Bonnie++. Available from http://www.coker.com.au/bonnie++/.

[48] Microsoft Corporation. Microsoft Windows access control model. Online at http://msdn.microsoft.com/en-us/library/aa374860%28VS.85%29.aspx.

[49] Jason Crampton, George Loizou, and Greg O' Shea. A logic of access control. *The Computer Journal*, 44(1):137–149, 2001.

[50] Haskell B. Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences*, 20:584–590, 1934.

[51] Marc Denecker and Antonis C. Kakas. Abduction in logic programming. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, pages 402–436. Springer-Verlag, 2002.

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

[53] Henry DeYoung. A logic for reasoning about time-dependent access control policies. Technical Report CMU-CS-08-131, Computer Science Department, Carnegie Mellon University, December 2008.

[54] 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)*, pages 133–145, June 2008. Extended version available as Carnegie Mellon University Technical Report CMU-CS-07-166.

[55] Henry DeYoung and Frank Pfenning. Reasoning about the consequences of authorization policies in a linear epistemic logic, 2009. Workshop on Foundations of Computer Security (FCS'09). Online at http://www.cs.cmu.edu/~hdeyoung/papers/fcs09.pdf.

[56] Nikhil Dinesh, Aravind Joshi, Insup Lee, and Oleg Sokolsky. Permission to speak: A logic for access control and conformance, 2008. Workshop on Formal Languages for Contract-Oriented Software (FLACOS'08). Online at http://www.cis.upenn.edu/~nikhild/permtospeak.pdf.

[57] Conal Elliott and Frank Pfenning. A semi-functional implementation of a higher-order logic programming language. In Peter Lee, editor, *Topics in Advanced Language Implementation*, pages 289–325. MIT Press, 1991.

[58] C.M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. See http://www.ietf.org/rfc/rfc2693.txt, 1999.

[59] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*. The MIT Press, 1990.

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

[61] Cédric Fournet, Andrew Gordon, and Sergio Maffeis. A type discipline for authorization in distributed systems. In *CSF '07: Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 31–48, 2007.

[62] Thom Frühwirth. Temporal annotated constraint logic programming. *Journal of Symbolic Computation*, 22(5-6):555–583, 1996.

[63] Deepak Garg. Principal-centric reasoning in constructive authorization logic, 2008. Workshop on Intuitionistic Modal Logic and Applications (IMLA'08). Full version available as Carnegie Mellon University Technical Report CMU-CS-09-120.

[64] Deepak Garg. Principal-centric reasoning in constructive authorization logic. Technical Report CMU-CS-09-120, Carnegie Mellon University, 2009.

[65] Deepak Garg and Martín Abadi. A modal deconstruction of access control logics. In *Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2008)*, pages 216–230, 2008.

[66] Deepak Garg, Lujo Bauer, Kevin Bowers, Frank Pfenning, and Michael Reiter. A linear logic of affirmation and knowledge. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS '06)*, pages 297–312, 2006.

[67] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In *Proceedings of the 19th Computer Security Foundations Workshop (CSFW '06)*, pages 283–293, 2006.

[68] Deepak Garg and Frank Pfenning. A proof-carrying file system. Technical Report CMU-CS-09-123, Carnegie Mellon University, 2009.

[69] Deepak Garg, Frank Pfenning, Denis Serenyi, and Brian Witten. A logical representation of common rules for controlling access to classified information. Technical Report CMU-CS-09-139, Carnegie Mellon University, 2009.

[70] 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.

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

[72] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, 1989.

[73] H. Gobioff, G. Gibson, and D. Tygar. Security for network attached storage devices. Technical Report CMU-CS-97-185, Carnegie Mellon University, 1997.

[74] Kurt Gödel. Eine Interpretation des intuitionistischen Aussagenkalkuls. *Ergebnisse eines mathematischen Kolloquiums*, 8:39–40, 1933.

[75] Timothy G. Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 47–58, 1990.

[76] Yuri Gurevich and Itay Neeman. DKAL: Distributed-knowledge authorization language. In *Proceedings of the 21st IEEE Symposium on Computer Security Foundations (CSF-21)*, pages 149–162, 2008.

[77] Christopher R. Hertel. *Implementing CIFS: The Common Internet File System*. Prentice Hall PTR, 2003.

[78] Joshua S. Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[79] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure. See http://www.ietf.org/rfc/rfc2459.txt, 1999.

[80] Jacob M. Howe. *Proof Search Issues in Some Non-Classical Logics*. PhD thesis, University of St Andrews, September 1998.

[81] John Ioannidis, Sotiris Ioannidis, Angelos Keromytis, and Vassilis Prevelakis. Fileteller: Paying and getting paid for file storage. In *Proceedings of the Sixth International Conference on Financial Cryptography*, pages 282–299, 2002.

[82] Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[83] Sushil Jajodia, Pierangela Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 31–42, 1997.

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

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

[86] Trevor Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 106–115, 2001.

[87] Michael Kaminsky, George Savvides, David Mazieres, and M. Frans Kaashoek. Decentralized user authentication in a global file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, pages 60–73, 2003.

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

[89] Butler W. Lampson. Protection. In *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, pages 437–443, 1971.

[90] 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)*, pages 432–444, 2007.

[91] Alexander Levine, Vassilis Prevelakis, John Ioannidis, Sotiris Ioannidis, and Angelos D. Keromytis. Webdava: An administrator-free approach to web file-sharing. In *WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies*, pages 59–64, 2003.

[92] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and Systems Security*, 6(1):128–171, 2003.

[93] Ninghui Li and Ziqing Mao. Administration in role-based access control. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 127–138, 2007.

[94] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73, 2003.

[95] Ninghui Li, John C. Mitchell, and W.H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[96] Ninghui Li, John C. Mitchell, and William H. Winsborough. Beyond proof-of-compliance: security analysis in trust management. *Journal of ACM*, 52(3):474–514, 2005.

[97] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, 2003.

[98] Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In *Proceedings of the 7th International Symposium on Principles and Practice of Declarative Programming (PPDP'05)*, pages 35–46, 2005.

[99] 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.

[100] David Mazières, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. *SIGOPS Operating Systems Review*, 34(2):19–20, 2000.

[101] Sean McLaughlin and Frank Pfenning. Efficient intuitionistic theorem proving with the polarized inverse method. In *Proceedings of the 22nd International Conference on Automated Deduction (CADE-22)*, pages 230–244, 2009.

[102] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[103] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[104] Stefan Miltchev, Vassilis Prevelakis, Sotiris Ioannidis, John Ioannidis, Angelos D. Keromytis, and Jonathan M. Smith. Secure and flexible global file sharing. In *Proceedings of the Annual USENIX Technical Conference, Freenix Track*, pages 165–178, 2003.

[105] Stefan Miltchev, Jonathan M. Smith, Vassilis Prevelakis, Angelos Keromytis, and Sotiris Ioannidis. Decentralized access control in distributed file systems. *ACM Computing Surveys*, 40(3):1–30, 2008.

[106] Tom Murphy, VII. *Modal Types for Mobile Code*. PhD thesis, Carnegie Mellon University, 2008. Available as technical report CMU-CS-08-126.

[107] OASIS. eXtensible Access Control Markup Language (XACML). Online at http://www.oasis-open.org/committees/xacml.

[108] Office of the Director of Central Intelligence. DCID 1/19: Security policy for sensitive compartmented information and security policy manual, 1995. Online at http://www.fas.org/irp/offdocs/dcid1-7.html.

[109] Office of the Director of Central Intelligence. DCID 1/7: Security controls on the dissemination of intelligence information, 1998. Online at http://www.fas.org/irp/offdocs/dcid1-19.html.

[110] Office of the Press Secretary of the White House. Executive order 12958: Classified national security information, 1995. Online at http://nsi.org/Library/Govt/ExecOrder12958.html.

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

[112] Christopher Olson and Ethan L. Miller. Secure capabilities for a petabyte-scale object-based distributed file system. In *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 64–73, 2005.

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

[114] Frank Pfenning. Automated theorem proving, 2004. Lecture notes for a class at Carnegie Mellon University. Available electronically from http://www.cs.cmu.edu/~fp/courses/atp.

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

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

[117] Benjamin C. Pierce and David N. Turner. Local type inference. *ACM Transactions on Programming Languages and Systems*, 22(1):1–44, 2000.

[118] Andrew Pimlott and Oleg Kiselyov. Soutei, a logic-based trust-management system. In *Proceedings of the Eighth International Symposium on Functional and Logic Programming (FLOPS 2006)*, pages 130–145, 2006.

[119] G. Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.

[120] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study.* Almquist and Wiksell, Stockholm, 1965.

[121] Benjamin C. Reed, Edward G. Chron, Randal C. Burns, and Darrell D. E. Long. Authenticating network-attached storage. *IEEE Micro*, 20(1):49–57, 2000.

[122] Jason Reed. Hybridizing a logical framework. In *International Workshop on Hybrid Logic 2006 (HyLo 2006)*, volume 174(6) of *Electronic Notes in Computer Science*, pages 135–148, June 2006.

[123] Jude T. Regan and Christian D. Jensen. Capability file names: Separating authorisation from user management in an internet file system. In *Proceedings of the 10th conference on USENIX Security Symposium*, pages 17–17, 2001.

[124] Peter Reiher, Thomas Page, Jr., Gerald Popek, Jeff Cook, and Stephen Crocker. Truffles - a secure service for widespread file sharing. In *Proceedings of the Privacy and Security Research Group Workshop on Network and Distributed System Security*, 1993.

[125] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh, and B. Lyon. *Design and implementation of the Sun network filesystem*, pages 379–390. Innovations in Internetworking. Artech House, Inc., 1988.

[126] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and Systems Security*, 2(1):105–135, 1999.

[127] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[128] Uluç Saranli and Frank Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems. In *Proceedings of the International Conference on Robotics and Automation (ICRA '07)*, pages 3705–3710, 2007.

[129] Amit Sasturkar, Ping Yang, Scott D. Stoller, and C.R. Ramakrishnan. Policy analysis for administrative role based access control. In *Proceedings of the 19th Computer Security Foundations Workshop*, pages 124–138. IEEE Computer Society Press, 2006.

[130] Mahadev Satyanarayanan. Scalable, secure, and highly available distributed file access. *Computer*, 23(5):9–18, 20–21, 1990.

[131] Andreas Schaad and Jonathan D. Moffett. A lightweight approach to specification and analysis of role-based access control extensions. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 13–22, 2002.

[132] Fred B. Schneider, Kevin Walsh, and Emin Gün Sirer. Nexus Authorization Logic (NAL): Design rationale and applications. Technical report, Cornell University, 2009. Online at http://ecommons.library.cornell.edu/handle/1813/13679.

[133] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck. Network File System (NFS) version 4 protocol, 2003. RFC 3050. Online at http://www.ietf.org/rfc/rfc3050.txt.

[134] IEEE Computer Society. *1003.1-2001 IEEE Information Technology - Portable Operating System Interface (POSIX.)*. IEEE Computer Society Press, 2001.

[135] Scott D. Stoller, Ping Yang, C R. Ramakrishnan, and Mikhail I. Gofman. Efficient policy analysis for administrative role based access control. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 445–455, 2007.

[136] D. B. Terry, M. M. Theimer, Karin Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP)*, pages 172–182, 1995.

[137] K. L. Thompson. UNIX implementation. *The Bell System Technical Journal*, 57(6):1931–1946, 1978.

[138] Amin Vahdat. *WebOS: Operating System Services for Wide Area Applications*. PhD thesis, University of California, Berkeley, 1997.

[139] Jeffrey A. Vaughan, Limin Jia, Karl Mazurak, and Steve Zdancewic. Evidence-based audit. In *Proceedings of the 21st IEEE Symposium on Computer Security Foundations (CSF-21)*, pages 177–191, 2008.

[140] Philip Wadler. A taste of linear logic. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS '93)*, pages 185–210, 1993.

[141] Philip Wadler. A syntax for linear logic. In *Proceedings of the 9th International Conference on Mathematical Foundations of Programming Semantics*, pages 513–529, 1994.

[142] Dan Williams, Patrick Reynolds, Kevin Walsh, Emin Gün Sirer, and Fred B. Schneider. Device driver safety through a reference validation mechanism. In *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)*, pages 241–254, 2008.

[143] Edward Wobber, Martín Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, 1994.

[144] D. A. Wright. Linear, strictness and usage logics. In *Proceedings of Conference on Computing: The Australian Theory Symposium*, pages 73–80, 1996.

[145] Jeffrey Zucker. The correspondence between cut-elimination and normalization. *Annals of Mathematical Logic*, 7:1–155, 1974.