Project Report
Categorical Judgments in a Logical Framework

15-816 Modal Logic
Henry DeYoung, hdeyoung@cs.cmu.edu

May 8, 2010

Abstract

Categorical judgments possess a context-clearing property, making them difficult to express elegantly in the LF logical framework because the context of available LF hypotheses grows monotonically. We describe a connection between categorical judgments and a refinement to open terms of LF's subordination relation. Leveraging this connection, we propose a logical framework, based on open-terms subordination, that supports elegant higher-order encodings of categorical judgments. As a concrete example of its expressive power, we present an encoding of judgmental S4 modal logic and establish its adequacy.

1 Introduction

The LF logical framework [HHP93] and its metalogic Twelf [PS99] have proven to be an extraordinarily useful methodology and system for the specification and metatheoretic analysis of logics and programming languages. Much of this power is derived from the elegant, higher-order way in which object-language hypothetical and parametric judgments are identified with the corresponding meta-language judgments. This typically leads to formalizations of object-language α-equivalence and substitution for "free," on the basis of the underlying meta-language operations.

At the same time, it is by no means a straightforward exercise to give adequate, higher-order LF encodings of deductive systems containing other classes of judgments which affect the context of available assumptions. For example, in judgmental S4 modal logic (JS4) [PD01], the validity judgment is categorical with respect to ordinary truth—proofs of validity may not

depend on truth assumptions. Because LF assumptions persist throughout the whole of a meta-language derivation, this context-clearing property is problematic when encoding categorical judgments such as JS4 validity.

Appearing to be quite similar to this property, that proofs of a given categorical judgment may not depend on certain kinds of assumptions, is the notion of subordination [Vir99]. Roughly, we say that an LF type family a is subordinate to an LF type family b if it is possible for an LF term of family a to appear inside a term of family b. Conversely, when a is *not* subordinate to b, we can be sure that terms of family b cannot contain terms of family a.

This project therefore aims to provide an account of the similarity between categorical judgments and (non-)subordination. The primary goal is to leverage this similarity to propose an extension of LF that admits adequate, higher-order encodings of categorical judgments.

In fact, as we describe in Section 4, categorical judgments do not correspond to the standard notion of subordination, but rather to a refined notion which deals specifically with *open* terms. As a proposed framework for encoding categorical judgments, in Section 6, we present $\mathrm{LF}^{\Box_{\preceq_o}}$, a version of LF with validity judgments, indexed by type families and related via the open-terms subordination preorder. Finally, in Section 8, we establish the adequacy of an encoding of JS4, thereby demonstrating the efficacy of the $\mathrm{LF}^{\Box_{\preceq_o}}$ framework.

Unfortunately, we fall short of our goal of giving an internally sound framework: we have not been able to prove (or disprove) a substitution theorem for $\mathrm{LF}^{\Box_{\preceq_o}}$. This is discussed in Section 7 in more detail, where we describe several proof attempts.

Organization of the Report. In Section 2, we review judgmental S4 modal logic, as it will serve as our motivating example. To better understand what goes wrong in encodings of JS4 in LF, we attempt several encodings in Section 3. In Section 4, we attempt an encoding with subordination built in to the LF type theory, the failure of which allows us to recognize the distinction between categorical judgments and subordination. Section 5 describes an intermediate framework based on open-terms subordination, but argues that the validity substitution principle of JS4 does not come for free. Sections 6 and 7 presents $\mathrm{LF}^{\Box_{\preceq_o}}$ and its metatheory. Section 8 establishes the adequacy of an encoding of JS4. Finally, Section 9 overviews related work.

$$
\begin{array}{lrcl}
\text{Propositions} & A, B, C & ::= & P \mid A \supset B \mid \Box A \\
\text{Truth Contexts} & \gamma & ::= & \cdot \mid \gamma, x{:}A\ \mathit{true} \\
\text{Validity Contexts} & \delta & ::= & \cdot \mid \delta, u{::}A\ \mathit{valid}
\end{array}
$$

Figure 1: Syntax of JS4

$\boxed{A\ \mathit{prop}}$

$$\frac{A\ \mathit{prop} \quad B\ \mathit{prop}}{A \supset B\ \mathit{prop}}\ \supset F \qquad\qquad \frac{A\ \mathit{prop}}{\Box A\ \mathit{prop}}\ \Box F$$

$\boxed{\gamma\ \mathsf{ctx}}$

$$\frac{}{\cdot\ \mathsf{ctx}}\ \mathsf{nil}_\gamma F \qquad\qquad \frac{\gamma\ \mathsf{ctx} \quad A\ \mathit{prop} \quad x \notin \mathrm{dom}\,\gamma}{(\gamma, x{:}A\ \mathit{true})\ \mathsf{ctx}}\ \mathsf{cons}_\gamma F$$

$\boxed{\delta\ \mathsf{vctx}}$

$$\frac{}{\cdot\ \mathsf{vctx}}\ \mathsf{nil}_\delta F \qquad\qquad \frac{\delta\ \mathsf{vctx} \quad A\ \mathit{prop} \quad u \notin \mathrm{dom}\,\delta}{(\delta, u{::}A\ \mathit{valid})\ \mathsf{vctx}}\ \mathsf{cons}_\delta F$$

Figure 2: Proposition and Context Formation for JS4

2 Review of JS4

Before diving into an attempt at encoding JS4 in LF, we would like to briefly review the implication and necessity fragment of judgmental S4 modal logic (JS4) [PD01].

Syntax and Formation Judgments. The syntax of JS4 is given in Figure 1. Propositions A, B, and C (and decorated variants) may be atomic, implication, or necessity. Because we will need to reason from assumptions, truth contexts γ and validity contexts δ are included. To reduce notational clutter, we will typically omit the judgment labels *true* and *valid* when they are apparent from the context.

Figure 2 gives the formation rules for JS4 propositions and contexts. The judgment $A\ \mathit{prop}$ checks the well-formedness of propositions. A proposition is well-formed when its constituents are. Judgments $\gamma\ \mathsf{ctx}$ and $\delta\ \mathsf{vctx}$

$\boxed{\delta; \gamma \vdash A\ true}$

$$\frac{}{\delta; \gamma, x{:}A\ true \vdash A\ true}\ x \qquad \frac{}{\delta, u{::}A\ valid; \gamma \vdash A\ true}\ u$$

$$\frac{\delta; \gamma, x{:}A\ true \vdash B\ true}{\delta; \gamma \vdash A \supset B\ true}\ \supset I^x \qquad \frac{\delta; \gamma \vdash A \supset B\ true \quad \delta; \gamma \vdash A\ true}{\delta; \gamma \vdash B\ true}\ \supset E$$

$$\frac{\delta; \cdot \vdash A\ true}{\delta; \gamma \vdash \Box A\ true}\ \Box I \qquad \frac{\delta; \gamma \vdash \Box A\ true \quad \delta, u{::}A\ valid; \gamma \vdash C\ true}{\delta; \gamma \vdash C\ true}\ \Box E^u$$

Figure 3: Inference Rules for JS4

verify that JS4 contexts are well-formed. These judgments hold if each hypothesis consists of a unique label and a well-formed proposition.

Judgmental Principles and Inference Rules. Because we will want to reason from hypotheses, the main form of judgment in JS4 is the hypothetical judgment $\delta; \gamma \vdash C\ true$. To ensure that JS4 respects the meaning of a hypothetical judgment, it must adhere to two principles: a hypothesis should suffice as evidence for a conclusion of the same (identity), and evidence for a conclusion should justify a hypothesis of the same in a proof (substitution).

The usual inference rules for truth hypotheses and implication are given in Figure 3.

The intended meaning of the categorical judgment $A\ valid$ is that A is *necessarily* true, that is, true without reliance on truth hypotheses. As a result, the following definition of validity is adopted:

Definition (Validity)**.**

1. *If* $\cdot \vdash A\ true$*, then* $A\ valid$*.*
2. *If* $A\ valid$*, then* $\gamma \vdash A\ true$*.*

The second part of this definition acts as the identity principle for validity. Based on this definition, the following substitution principle is obtained:

Principle (Substitution for Validity)**.**

If $\delta; \cdot \vdash A\ true$ *and* $\delta, u{::}A\ valid; \gamma \vdash C\ true$*, then* $\delta; \gamma \vdash C\ true$*.*

With these principles in hand, the inference rules for validity (see Figure 3) may be justified. The identity principle is captured as a hypothesis rule, labeled with the name of the corresponding hypothesis. The proposition $\Box A$ internalizes validity, so it is introduced in the $\Box I$ rule by establishing the corresponding validity judgment. (The first part of the definition of validity is silently applied in this premise.) Finally, the $\Box E^u$ rule eliminates necessity in what is essentially an application of the substitution principle for validity.

3 Attempting an Encoding of JS4 in LF

To motivate our desire to develop a method for encoding categorical judgments in an LF-style logical framework, we now turn to describing several unsuccessful attempts at encoding the implication and necessity fragment of JS4 in LF. This will also serve as a means of discovering the expressive power that is needed in our extension of LF.

Since the syntax of JS4 is not fundamentally different than that of implicational propositional logic, we expect that a standard LF encoding of propositions will carry over to JS4:

$$\begin{aligned}&\texttt{o} : \mathsf{type}.\\ &\texttt{imp} : \texttt{o} \to \texttt{o} \to \texttt{o}.\\ &\texttt{box} : \texttt{o} \to \texttt{o}.\end{aligned}$$

We also expect that the encoding of the implicational fragment should remain the same:

$$\begin{aligned}&\texttt{true} : \texttt{o} \to \mathsf{type}.\\ &\texttt{impI} : \Pi A{:}\texttt{o}.\, \Pi B{:}\texttt{o}.\, (\texttt{true}\ A \to \texttt{true}\ B) \to \texttt{true}\ (\texttt{imp}\ A\ B).\\ &\texttt{impE} : \Pi A{:}\texttt{o}.\, \Pi B{:}\texttt{o}.\, \texttt{true}\ (\texttt{imp}\ A\ B) \to \texttt{true}\ A \to \texttt{true}\ B.\end{aligned}$$

The most natural attempt at encoding the necessity fragment would be to mimic the judgmental apparatus of JS4 and include the type family

$$\texttt{lvalid} : \texttt{o} \to \mathsf{type}.$$

where the name `lvalid` is intended to suggest that this family should only be used on the left of LF hypothetical judgments.

In LF, there is no support for subtyping. Therefore, given this approach, we will need an explicit coercion from `lvalid` to `true` to serve as the encoding of the validity hypothesis rule. We contend that this rule and the

$\Box E^u$ rule are straightforward:

$$\texttt{vhyp} : \Pi A{:}\texttt{o}.\, \texttt{lvalid}\ A \to \texttt{true}\ A.$$
$$\texttt{boxE} : \Pi A{:}\texttt{o}.\, \Pi C{:}\texttt{o}.\, \texttt{true}\ (\texttt{box}\ A) \to (\texttt{lvalid}\ A \to \texttt{true}\ C) \to \texttt{true}\ C.$$

The question is what encoding should be given for the $\Box I$ rule.

Attempt 1: In strict syntactic correspondence with JS4, one might initially consider a constant:

$$\texttt{boxI1} : \Pi A{:}\texttt{o}.\, \texttt{true}\ A \to \texttt{true}\ (\texttt{box}\ A).$$

Unfortunately, without any additional restrictions, such an encoding would be unsound with respect to JS4. For example, from $x{:}\texttt{true}\ A$ we could construct a term of type $\texttt{true}\ (\texttt{box}\ A)$, parametrically in A; truth and validity have incorrectly collapsed within the encoding. In other words, this encoding is too permissive in that it fails to prevent truth hypotheses from occurring under $\Box$s.

Attempt 2a: Next, one might recall that A *valid* is defined as $\cdot \vdash A$ *true* in JS4. Since `lvalid` is the family corresponding to the validity judgment, perhaps the $\Box I$ rule could instead be encoded with a constant:

$$\texttt{boxI2} : \Pi A{:}\texttt{o}.\, \texttt{lvalid}\ A \to \texttt{true}\ (\texttt{box}\ A).$$

Though this encoding is sound, it is also incomplete. For example, one cannot construct a term of type $\Pi A{:}\texttt{o}.\, \texttt{true}\ (\texttt{box}\ (\texttt{imp}\ A\ A))$ in the empty LF context. In other words, this encoding is too restrictive in that there is no way to build terms of family `lvalid` which are not variables.

Attempt 2b: In an attempt to find a midpoint between these extremes, one might consider introducing a constant corresponding to the definition of validity:

$$\texttt{vconc2} : \Pi A{:}\texttt{o}.\, \texttt{true}\ A \to \texttt{lvalid}\ A.$$

In addition, the `boxI2` constant would be retained. This resolves the incompleteness problem. However, without any additional restrictions, we are now back to an unsound encoding. The term $\texttt{boxI2}\ A\ (\texttt{vconc2}\ A\ M)$ is morally the same as $\texttt{boxI1}\ A\ M$, and so the counterexample from the first attempt still applies.

Moreover, there is a new problem related to the appearance of `lvalid` on the right of the LF hypothetical judgment. Under the hypothesis that

x_u:`lvalid` A, both `boxI2` A x_u and `boxI2` A (`vconc2` A x_u) would be terms of type `true` (`box` A) intended to correspond to the JS4 derivation

$$\dfrac{\dfrac{}{\delta, u{::}A; \cdot \vdash A}\, u}{\delta, u{::}A; \gamma \vdash \Box A}\, \Box I$$

As a result, we will be unable to obtain a bijection between JS4 derivations and LF terms of family `true` under this encoding.

Attempt 3: A possible fix for this new problem is to introduce a type family `rvalid`, so named because it only appears on the right. Then, the `vconc` and `boxI` constants would be revised to reflect the change:

$$\texttt{rvalid} : \texttt{o} \rightarrow \mathsf{type}.$$
$$\texttt{vconc3} : \Pi A{:}\texttt{o}.\, \texttt{true}\ A \rightarrow \texttt{rvalid}\ A.$$
$$\texttt{boxI3} : \Pi A{:}\texttt{o}.\, \texttt{rvalid}\ A \rightarrow \texttt{true}\ (\texttt{box}\ A).$$

There is now a one-to-one correspondence between LF terms and JS4 derivations because there are no LF hypotheses of family `rvalid`. However, the unsoundness of the encoding for the $\Box I$ rule persists.

4 Attempting an Encoding of JS4 in LF$^{\preceq}$

The key problem with our first and third attempts at an encoding of JS4 in LF was the absence of a method for clearing the truth hypotheses before proving the $\Box I$ rule's premise.

As previously noted, the notion of subordination developed by Virga [Vir99] appears to be similar to a context-clearing effect. A type family a is subordinate to type family b, written $a \preceq b$, if terms of family a may appear in types or terms of family b. Conversely, when $a \not\preceq b$, terms of family a may not appear in types or terms of family b.

Leveraging this apparent similarity, a possible idea is to prescribe a constraint on the subordination relation, such as `true` $\not\preceq$ `rvalid`, that might capture the context-clearing nature of the categorical judgment. The hope is that one could then obtain an adequate encoding of the $\Box I$ rule in LF$^{\preceq}$ [Vir99][1], a version of LF that supports subordination in the type theory.

Unfortunately, upon closer examination, subordination and categorical judgments are not quite the same. Non-subordination demands that *all*

[1] Refer to [HL07] for a modern canonical forms presentation of LF$^{\preceq}$.

terms of a given family, whether open or closed, not appear in types or terms of another family. For instance, in LF$^{\preceq}$, the constant

$$\texttt{vconc3} : \Pi A{:}\texttt{o}.\, \texttt{true}\ A \rightarrow \texttt{rvalid}\ A.$$

would be rejected when $\texttt{true} \not\preceq \texttt{rvalid}$. This is because admitting such a constant would allow terms of family `true` to appear in terms of family `rvalid`, such as $\texttt{impI}\ A\ A\ (\lambda y.\, y)$ in $\texttt{vconc3}\ A\ (\texttt{impI}\ A\ A\ (\lambda y.\, y))$.

On the other hand, a categorical judgment demands that only the *open* derivations of a given judgment not appear. For example, consider the derivations:

$$\mathcal{D}_1 = \dfrac{\dfrac{\dfrac{}{\cdot; y{:}A \vdash A\ \mathit{true}}\, y}{\cdot; \cdot \vdash A \supset A\ \mathit{true}}\, \supset I^y}{\cdot; x{:}A \supset A \vdash \Box(A \supset A)\ \mathit{true}}\, \Box I \qquad \mathcal{D}_2 = \dfrac{\dfrac{}{u{::}A; \cdot \vdash A\ \mathit{true}}\, u}{u{::}A; x{:}A \vdash \Box A\ \mathit{true}}\, \Box I$$

$$\mathcal{D}_3 = \dfrac{\dfrac{}{u{::}A; x{:}A \vdash A\ \mathit{true}}\, x}{u{::}A; x{:}A \vdash \Box A\ \mathit{true}}\, ?$$

The derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ are indeed well-formed because their subderivations of $A \supset A\ \mathit{true}$ and $A\ \mathit{true}$, respectively, are closed with respect to truth hypotheses. (Note, however, that the subderivation may be open with respect to validity hypotheses, as $\mathcal{D}_2$ demonstrates.) On the other hand, the derivation $\mathcal{D}_3$ is not well-formed because its subderivation of $A\ \mathit{true}$ is open with respect to $x{:}A\ \mathit{true}$.

Based on this distinction, we contend that the standard notion of all-terms subordination is too restrictive for our purposes. Requiring $\texttt{true} \not\preceq \texttt{rvalid}$, disallows truth-closed terms needed for adequacy, such as

$$\texttt{boxI3}\ A\ (\texttt{vconc3}\ A\ (\texttt{impI}\ A\ A\ (\lambda y.\, y))),$$

which we expect to be the encoding of $\mathcal{D}_1$. We therefore propose a second, refined notion of *open-terms* subordination. We shall say that a type family a is open-subordinate to a type family b, written $a \preceq_o b$, if *open* terms of family a may appear in types or terms of family b. Conversely, when $a \not\preceq_o b$, open terms of family a may not appear in types or terms of family b.

5 Attempting an Encoding with Open-Terms Subordination

Given a notion of open-terms subordination, one idea is to design a version of LF where open-terms subordination is built into the type theory, similar to the way that all-terms subordination is built into LF$^{\preceq}$. We attempted such a framework, which we call LF$^{\preceq_o}$. Its defining characteristic is that hypotheses not open-subordinate to some family a are removed from the ambient context when a term is checked against a type of family a. This is best exemplified by the second premise of the rule

$$\frac{\Gamma \vdash R \Rightarrow \Pi x{:}A_2.\, A \quad \Gamma|^{\preceq_o}_{A_2} \vdash M \Leftarrow A_2}{\Gamma \vdash R\ M \Rightarrow [M/x]A}$$

where $\Gamma|^{\preceq_o}_{A_2}$ is the result of removing from Γ all hypothesis that are not open-subordinate to the family to which type A_2 belongs. By removing these hypotheses, no open terms of a family not open-subordinate to the family of A_2 may appear in well-typed M: all variables needed to build such open terms are no longer in scope.

Let us return to our third attempt at an encoding of JS4, examining it in the context of this framework based on open-terms subordination.

$\texttt{true} : \texttt{o} \rightarrow \mathsf{type}.$
$\texttt{lvalid} : \texttt{o} \rightarrow \mathsf{type}.$
$\texttt{rvalid} : \texttt{o} \rightarrow \mathsf{type}.$

$\texttt{true} \not\preceq_o \texttt{rvalid}. \quad \texttt{lvalid} \preceq_o \texttt{rvalid}.$
$\texttt{true} \preceq_o \texttt{true}. \quad \texttt{lvalid} \preceq_o \texttt{true}.$

$\texttt{vhyp} : \Pi A{:}\texttt{o}.\, \texttt{lvalid}\ A \rightarrow \texttt{true}\ A.$
$\texttt{vconc3} : \Pi A{:}\texttt{o}.\, \texttt{true}\ A \rightarrow \texttt{rvalid}\ A.$
$\texttt{boxI3} : \Pi A{:}\texttt{o}.\, \texttt{rvalid}\ A \rightarrow \texttt{true}\ (\texttt{box}\ A).$
$\texttt{boxE} : \Pi A{:}\texttt{o}.\, \Pi C{:}\texttt{o}.\, \texttt{true}\ (\texttt{box}\ A) \rightarrow (\texttt{lvalid}\ A \rightarrow \texttt{true}\ C) \rightarrow \texttt{true}\ C.$

In the fourth line, we now prescribe that `true` is not open-subordinate to `rvalid`. Thus, when type checking a term $\texttt{boxI3}\ A\ (\texttt{vconc3}\ A\ M)$, the term M will be checked in a context that does not contain hypotheses of family `true`:

$$\Gamma|^{\preceq_o}_{\texttt{rvalid}}|^{\preceq_o}_{\texttt{true}} \vdash M \Leftarrow \texttt{true}\ A.$$

Because $\texttt{true} \not\preceq_o \texttt{rvalid}$ and $\texttt{lvalid} \preceq_o \texttt{rvalid}$, the innermost context restriction, $|^{\preceq_o}_{\texttt{rvalid}}$, removes from Γ all hypotheses of family `true` but retains

all hypotheses of family `lvalid`. Thus, by this context restriction, the $\Box I$ rule's premise is now adequately encoded.

5.1 The Substitution Principle for Validity is Not Free

The traditional advantages of higher-order encodings in LF are that object-language α-equivalence and substitution principles can be obtained for "free", by using the corresponding notions from the LF meta-language notions. By using our $\text{LF}^{\preceq_o}$-level notion of context restriction, we have similarly obtained context-clearing for "free".

Unfortunately, under our encoding, it does not seem possible to obtain the substitution principle for validity (see Section 2) for free in $\text{LF}^{\preceq_o}$. We would need to express a substitution of some $\text{LF}^{\preceq_o}$ term for a hypothesis x_u:`lvalid` A. As there are no constants for constructing terms of family `lvalid`, such a substitution would essentially be an α-variation and would not capture the desired substitution principle.

Failure to obtain the validity substitution principle for free is a consequence of having separate type families for validity hypotheses (`lvalid`) and conclusions (`rvalid`).[2] This is similar to the failure to obtain a cut principle for free in the sequent calculus since separate hypothesis and conclusion judgments are used there.

In the author's opinion, the absence of a free substitution principle for validity is not of major concern: it should be straightforward enough to prove this as a metatheorem in $\text{LF}^{\preceq_o}$. Nonetheless, as we show in the subsequent sections of this report, by adding meta-level validity hypotheses, it is possible to construct a more expressive framework in which the substitution principle for validity can indeed be obtained for free. Since we will argue that $\text{LF}^{\preceq_o}$ is a fragment of this framework, this approach avoids taking a strong position on the issue. If the user insists on having the substitution principle for free, that is possible; otherwise, the $\text{LF}^{\preceq_o}$ fragment can be used.

6 $\text{LF}^{\Box_{\preceq_o}}$

In keeping with the previous discussion, we propose a logical framework, called $\text{LF}^{\Box_{\preceq_o}}$, containing a family of validity judgments indexed by type

[2] It might appear that replacing occurrences of `rvalid` with `lvalid` would resolve this. However, doing so would be analogous to our second attempt at an encoding of JS4 in ordinary LF, and the problems with adequacy would carry over (see Section 3).

Kinds	K	$::=$	$\mathsf{type} \mid \Pi u{::}A[a].\, K$
Atomic Type Families	P	$::=$	$a \mid P\ N$
Canonical Types	A, B	$::=$	$P \mid \Pi u{::}A[a].\, B$
Atomic Terms	R	$::=$	$c \mid u \mid R\ N$
Canonical Terms	N, M	$::=$	$R \mid \lambda u.N$
Signatures	Σ	$::=$	$\cdot \mid \Sigma, a{:}K \mid \Sigma, c{:}A$
Contexts	Δ	$::=$	$\cdot \mid \Delta, u{::}A[a]$

Figure 4: Syntax of $\mathrm{LF}^{\Box_{\preceq_o}}$

family constants.

Validity will indicate respect for open-terms subordination. For instance, in the hypothesis $u{::}A[a]$, we intend that u stands for a term of type A that does not contain open terms of types not open-subordinate to type family a. To keep encodings clean, we want every hypothesis to carry a type family label. We therefore reject ordinary, non-valid typing hypotheses, instead relying only on the validity hypotheses.

6.1 Syntax

In Figure 4, we present the syntax of $\mathrm{LF}^{\Box_{\preceq_o}}$. Following the canonical forms approach to LF pioneered by Watkins *et al.* [WCPW02], we classify types and terms as either atomic or canonical.

Rather than internalizing family-indexed validity as a family-indexed necessity modality, we internalize the hypothetical judgment as the dependent modal function type $\Pi u{::}A[a].\, B$. This technique avoids commuting conversions and related "exotic" canonical terms, and is borrowed from the dependent contextual modal type theory [NPP08]. Because the framework rejects non-valid hypotheses, there is no ordinary dependent function type. Consequently, $\Pi u{::}A[a].\, B$ is the cornerstone of $\mathrm{LF}^{\Box_{\preceq_o}}$.

Just as we have dependent modal function types, we have corresponding dependent modal function kinds $\Pi u{::}A[a].\, K$. Again, there are no ordinary dependent function kinds, such as would be found in LF.

Since we have only modal function types and kinds, applications $R\ N$ and $P\ N$ at the level of terms and type families refer to *modal* applications. (We choose not to use decorations in the concrete syntax to make this explicit; juxtaposition is far too convenient concrete syntax to surrender.)

The remaining syntax is analogous to that of LF: a and b stand for type family constants, c stands for term constants, and u and v stand for variables.

6.2 Open-Terms Subordination

As previously motivated, we intend that a type family a is open-subordinate to a type family b, written $a \preceq_o b$, if terms of family a are permitted to appear in types or terms of family b. Formally, we impose the following requirements:

Definition (Open-Terms Subordination Relation)**.** *An* open-terms subordination relation $\preceq_o$ *for a signature* Σ *is a binary relation between family-level constants declared in* Σ *that satisfies:*

1. *Well-formedness: The judgment* $\vdash_{\preceq_o} \Sigma$ `sig` *is derivable (see Fig. 5).*
2. *Refinement: If* $\preceq$ *is an all-terms subordination relation for* Σ*, then* $\preceq_o \subseteq \preceq$*.*
3. *Reflexivity: For all* $a \in \mathrm{dom}\,\Sigma$*,* $a \preceq_o a$*.*
4. *Transitivity: If* $a_1 \preceq_o a_2$ *and* $a_2 \preceq_o a_3$*, then* $a_1 \preceq_o a_3$*.*

The well-formedness constraint enforces consistency of the signature with the open-terms subordination relation. The refinement property is necessary because it is incoherent to allow open terms to appear if no terms, whether open or closed, may appear. Reflexivity reflects the fact that an open term appears in itself, a term in its own right. The transitivity property is a result of the underlying "may appear in" relation naturally being transitive.

Because the open-terms subordination relation is given as a list of pairs of type family constants, decidability is trivial. This justifies use of the notation $a \not\preceq_o b$ to indicate that a is not open-subordinate to b.

Thus far, we have not imposed any conditions on open-terms subordination relations that suggest their connection to open terms. As previously sketched, this is accomplished by restriction of a context to a type family:

$\boxed{\Delta|_b^{\preceq_o} = \Delta'}$

$$\begin{array}{rcll} (\cdot)|_b^{\preceq_o} & = & \cdot & \\ (\Delta, u{::}A[a])|_b^{\preceq_o} & = & \Delta|_b^{\preceq_o}, u{::}A[a] & \text{if } a \preceq_o b \\ (\Delta, u{::}A[a])|_b^{\preceq_o} & = & \Delta|_b^{\preceq_o} & \text{if } a \not\preceq_o b \end{array}$$

In LF$^{\Box_{\preceq_o}}$, the hypothesis $u{::}A[a]$ acts as a placeholder for a term that is potentially open with respect to type families a' such that $a' \preceq_o a$. If $a \preceq_o b$,

$\boxed{\vdash_{\preceq_o} \Sigma\ \mathtt{sig}}$

$$\frac{}{\vdash_{\preceq_o} \cdot\ \mathtt{sig}}\ \text{SIG\_NIL} \qquad \frac{\vdash_{\preceq_o} \Sigma\ \mathtt{sig} \quad \cdot \vdash_{\Sigma,\preceq_o} K\ \mathsf{kind}}{\vdash_{\preceq_o} \Sigma, a{:}K\ \mathtt{sig}}\ \text{SIG\_CONS\_FAM}$$

$$\frac{\vdash_{\preceq_o} \Sigma\ \mathtt{sig} \quad \cdot \vdash_{\Sigma,\preceq_o} A\ \mathsf{type}}{\vdash_{\preceq_o} \Sigma, c{:}A\ \mathtt{sig}}\ \text{SIG\_CONS\_TM}$$

$\boxed{\vdash_{\Sigma,\preceq_o} \Delta\ \mathtt{vctx}}$

$$\frac{}{\vdash_{\Sigma,\preceq_o} \cdot\ \mathtt{vctx}}\ \text{VCTX\_NIL}$$

$$\frac{\vdash_{\Sigma,\preceq_o} \Delta\ \mathtt{vctx} \quad a \in \mathrm{dom}\,\Sigma \quad \Delta|_a^{\preceq_o} \vdash_{\Sigma,\preceq_o} A\ \mathsf{type}}{\vdash_{\Sigma,\preceq_o} \Delta, u{::}A[a]\ \mathtt{vctx}}\ \text{VCTX\_CONS}$$

Figure 5: Signature and Context Formation

then u may survive the restriction since all families a' that may appear in a term substituted for u are themselves open-subordinate to b (by transitivity). In this way, judicious use of a context restriction in the typing rules will ensure that open-terms subordination lives up to its intended meaning.

6.3 Signature and Context Well-Formedness

In Figure 5, we present the rules for checking well-formedness of signatures and contexts. Except for the last two premises of the VCTX_CONS rule, these rules are standard.

In the VCTX_CONS rule, the second premise checks that the judgment index a is truly a declared type family constant. The third premise checks that A is a well-formed type in the *restricted* context $\Delta|_a^{\preceq_o}$, rather than the full preceding context. This guarantees that context restriction preserves well-formedness, as we will prove in Section 7.

6.4 Typing Judgments

In analogy with LF, we have five key typing judgments:

$\Delta \vdash_{\Sigma,\preceq_o} K$ kind	K is checked to be a well-formed kind
$\Delta \vdash_{\Sigma,\preceq_o} P \Rightarrow K$	Kind K is synthesized for atomic type family P
$\Delta \vdash_{\Sigma,\preceq_o} A$ type	A is checked to be a well-formed canonical type
$\Delta \vdash_{\Sigma,\preceq_o} R \Rightarrow A$	Canonical type A is synthesized for atomic term R
$\Delta \vdash_{\Sigma,\preceq_o} N \Leftarrow A$	Canonical term N is checked to have canonical type A

In all judgments, the signature Σ, the open-terms subordination relation $\preceq_o$, the context Δ, and the subjects (K, P, A, R, and N, respectively) are inputs. In the term checking judgment, A is also an input. In the synthesis judgments, the remaining components, K and A, respectively, are outputs. Non-subject inputs are always presupposed to be well-formed, whereas outputs are guaranteed to be well-formed.

The rules for these judgments are presented in Figure 6. We describe the salient rules in detail.

ΠE **Rule:** To synthesize a canonical type for the modal function application $R\ N$, we first synthesize a canonical type for R. Because R is a function, the synthesized type should be a dependent modal function type $\Pi u{::}A_2[a].\, A$. Consequently, the argument N should have type A_2 and be valid with respect to type family a. Because a-validity means that N should be closed with respect to families not open-subordinate to a, the second premise utilizes a context restriction. This removes all hypotheses not open-subordinate to a, preventing them from appearing in N. We then check that N has type A_2.

As u may appear free in the type A, the synthesized type should be morally $[\![N/u]\!]A$, the modal substitution of N for u in A. Unfortunately, ordinary modal substitution may not produce a canonical type. Therefore, we follow Watkins *et al.* [WCPW02] and use a hereditary substitution, $[\![N/u]\!]^a_{A_2^-[a]}A$, which contracts any β-redices that would be introduced during ordinary substitution. We define hereditary substitution in the following section.

$\Pi_K E$ **Rule:** This rule is analogous to the ΠE rule. Because $\Pi u{::}A[a].\, K$ is the kind synthesized for P, the argument N should have type A and be a-valid. As in the ΠE rule, the second premise therefore checks N against type A in the restricted context $\Delta|^{\preceq_o}_a$.

$\boxed{\Delta \vdash_{\Sigma,\preceq_o} K \ \mathsf{kind}}$

$$\frac{}{\Delta \vdash \mathsf{type}\ \mathsf{kind}}\ \mathsf{type}F$$

$$\frac{a \in \mathrm{dom}\,\Sigma \quad \Delta|_a^{\prec_o} \vdash A\ \mathsf{type} \quad \Delta, u{::}A[a] \vdash K\ \mathsf{kind}}{\Delta \vdash \Pi u{::}A[a].\, K\ \mathsf{kind}}\ \Pi_K F$$

$\boxed{\Delta \vdash_{\Sigma,\preceq_o} P \Rightarrow K}$

$$\frac{a \in \mathrm{dom}\,\Sigma}{\Delta \vdash a \Rightarrow \Sigma(a)}\ a$$

$$\frac{\Delta \vdash P \Rightarrow \Pi u{::}A[a].\, K \quad \Delta|_a^{\preceq_o} \vdash N \Leftarrow A \quad [\![N/u]\!]^k_{A^-[a]} K = K'}{\Delta \vdash P\ N \Rightarrow K'}\ \Pi_K E$$

$\boxed{\Delta \vdash_{\Sigma,\preceq_o} A\ \mathsf{type}}$

$$\frac{a \in \mathrm{dom}\,\Sigma \quad \Delta|_a^{\prec_o} \vdash A_2\ \mathsf{type} \quad \Delta, u{::}A_2[a] \vdash A\ \mathsf{type}}{\Delta \vdash \Pi u{::}A_2[a].\, A\ \mathsf{type}}\ \Pi F$$

$$\frac{\Delta \vdash P \Rightarrow \mathsf{type}}{\Delta \vdash P\ \mathsf{type}}\ \Rightarrow\mathsf{type}$$

$\boxed{\Delta \vdash_{\Sigma,\preceq_o} R \Rightarrow A}$

$$\frac{c \in \mathrm{dom}\,\Sigma}{\Delta \vdash c \Rightarrow \Sigma(c)}\ c \qquad \frac{u \in \mathrm{dom}\,\Delta}{\Delta \vdash u \Rightarrow \Delta(u)}\ u$$

$$\frac{\Delta \vdash R \Rightarrow \Pi u{::}A_2[a].\, A \quad \Delta|_a^{\prec_o} \vdash N \Leftarrow A_2 \quad [\![N/u]\!]^a_{A_2^-[a]} A = A'}{\Delta \vdash R\ N \Rightarrow A'}\ \Pi E$$

$\boxed{\Delta \vdash_{\Sigma,\preceq_o} N \Leftarrow A}$

$$\frac{\Delta, u{::}A_2[a] \vdash N \Leftarrow A}{\Delta \vdash \lambda u.N \Leftarrow \Pi u{::}A_2[a].\, A}\ \Pi I \qquad \frac{\Delta \vdash R \Rightarrow P}{\Delta \vdash R \Leftarrow P}\ \Rightarrow\Leftarrow$$

Figure 6: LF$^{\Box_{\preceq_o}}$ typing rules.

ΠF **Rule:** Much of this rule is standard: to check that $\Pi u{::}A_2[a].\, A$ is a well-formed type, we check that A_2 is a well-formed type and then check that A is a well-formed type in the context extended with $u{::}A_2[a]$.

The main novelty is the context that we use when checking the well-formedness of A_2. By using the restricted context, $\Delta|_a^{\preceq_o}$, rather than the full context, we can maintain our invariant on the well-formedness of the type used in type checking. Specifically, from the first premise of the ΠE rule, we learn that $\Delta \vdash \Pi u{::}A_2[a].\, A$ type. Before making the call to the second premise of that rule, we must guarantee that $\Delta|_a^{\preceq_o} \vdash A_2$ type. Because we have designed the ΠF rule to use the restricted context, this indeed holds.

$\Pi_K F$ **Rule:** In analogy with the ΠF rule, we check the well-formedness of type A under the restricted context $\Delta|_a^{\preceq_o}$.

6.5 Hereditary Substitution

Since the syntax of terms precludes the existence of β-redices, we must be sure that no β-redices arise during typing. Ordinary substitution would violate this syntax. For example, although both $\lambda x.\, y\ x$ and $u\ z$ are β-normal, the substitution

$$[(\lambda x.\, y\ x)/u](u\ z) = (\lambda x.\, y\ x)\ z$$

is not. For this reason, we follow Watkins *et al.* [WCPW02] and use a hereditary substitution, which eliminates any β-redices that would be introduced during a standard substitution. For example, hereditary substitution of $\lambda x.\, y\ x$ for u in $u\ z$ would instead result in $y\ z$. In Figures 7 and 8, we present the definition of hereditary substitution of canonical terms in the various syntactic categories.

As typical, we annotate hereditary substitutions with the simple type of the term which is being substituted. Simple types, which are presented in Figure 9 ignore the dependent nature of types. This is captured by the erasure, $A^- = \alpha$, of a type A to its unique simple type α.

7 Metatheory of LF$^{\Box_{\preceq_o}}$

To gain confidence in the coherence of our design of LF$^{\Box_{\preceq_o}}$, we turn to a study of its metatheoretic properties. We are primarily interested in verifying that the typing judgments respect the meaning of a hypothetical judgment. Specifically, LF$^{\Box_{\preceq_o}}$ should enjoy substitution and identity principles:

$\boxed{[\![N/u]\!]^k_{\alpha[a]} K = K'}$

$$\frac{}{[\![N/u]\!]^k_{\alpha[a]} \mathsf{type} = \mathsf{type}}\ \text{SUBST\_K\_TYPE}$$

$$\frac{[\![N/u]\!]^a_{\alpha[a]} B_2 = B_2' \quad [\![N/u]\!]^k_{\alpha[a]} K = K'}{[\![N/u]\!]^k_{\alpha[a]} (\Pi v{::}B_2[b].\, K) = \Pi v{::}B_2'[b].\, K'}\ \text{SUBST\_K\_PI}$$

$\boxed{[\![N/u]\!]^p_{\alpha[a]} P = P'}$

$$\frac{}{[\![N/u]\!]^p_{\alpha[a]} b = b}\ \text{SUBST\_P\_CONST}$$

$$\frac{[\![N/u]\!]^p_{\alpha[a]} P = P' \quad [\![N/u]\!]^n_{\alpha[a]} M = M'}{[\![N/u]\!]^p_{\alpha[a]} (P\ M) = P'\ M'}\ \text{SUBST\_P\_APP}$$

$\boxed{[\![N/u]\!]^a_{\alpha[a]} A = A'}$

$$\frac{[\![N/u]\!]^p_{\alpha[a]} P = P'}{[\![N/u]\!]^a_{\alpha[a]} P = P'}\ \text{SUBST\_A\_P} \qquad \frac{[\![N/u]\!]^a_{\alpha[a]} B_2 = B_2' \quad [\![N/u]\!]^a_{\alpha[a]} B = B'}{[\![N/u]\!]^a_{\alpha[a]} (\Pi v{::}B_2[b].\, B) = \Pi v{::}B_2'[b].\, B'}\ \text{SUBST\_A\_PI}$$

$\boxed{[\![N/u]\!]^{rr}_{\alpha[a]} R = R'}$

$$\frac{}{[\![N/u]\!]^{rr}_{\alpha[a]} c = c}\ \text{SUBST\_RR\_CONST} \qquad \frac{v \neq u}{[\![N/u]\!]^{rr}_{\alpha[a]} v = v}\ \text{SUBST\_RR\_VAR}$$

$$\frac{[\![N/u]\!]^{rr}_{\alpha[a]} R = R' \quad [\![N/u]\!]^n_{\alpha[a]} M = M'}{[\![N/u]\!]^{rr}_{\alpha[a]} (R\ M) = R'\ M'}\ \text{SUBST\_RR\_APP}$$

$\boxed{[\![N/u]\!]^{rn}_{\alpha[a]} R = M : \alpha'}$

$$\frac{}{[\![N/u]\!]^{rn}_{\alpha[a]} u = N : \alpha}\ \text{SUBST\_RN\_VAR}$$

$$\frac{[\![N/u]\!]^{rn}_{\alpha[a]} R = \lambda v.N_1 : \beta_2[b] \to \beta \quad [\![N/u]\!]^n_{\alpha[a]} M = M' \quad [\![M'/v]\!]^n_{\beta_2[b]} N_1 = N_2}{[\![N/u]\!]^{rn}_{\alpha[a]} (R\ M) = N_2 : \beta}\ \text{SUBST\_RN\_APP}$$

Figure 7: Hereditary substitution for $\text{LF}^{\Box}_{\preceq_o}$

$\boxed{[\![N/u]\!]^n_{\alpha[a]} M = M'}$

$$\frac{[\![N/u]\!]^{rn}_{\alpha[a]} R = M' : \alpha'}{[\![N/u]\!]^n_{\alpha[a]} R = M'}\ \text{SUBST\_N\_RN} \qquad \frac{[\![N/u]\!]^{rr}_{\alpha[a]} R = R'}{[\![N/u]\!]^n_{\alpha[a]} R = R'}\ \text{SUBST\_N\_RR}$$

$$\frac{[\![N/u]\!]^n_{\alpha[a]} M = M'}{[\![N/u]\!]^n_{\alpha[a]} (\lambda v.M) = \lambda v.M'}\ \text{SUBST\_N\_LAM}$$

$\boxed{[\![N/u]\!]^c_{\alpha[a]} \Delta = \Delta'}$

$$\frac{}{[\![N/u]\!]^c_{\alpha[a]} \cdot = \cdot}\ \text{SUBST\_C\_NIL}$$

$$\frac{[\![N/u]\!]^c_{\alpha[a]} \Delta = \Delta' \quad [\![N/u]\!]^a_{\alpha[a]} B = B' \quad v \neq u \quad v \notin \mathrm{FV}(N)}{[\![N/u]\!]^c_{\alpha[a]} (\Delta, v{::}B[b]) = \Delta', v{::}B'[b]}\ \text{SUBST\_C\_CONS}$$

Figure 8: Hereditary substitution for $\mathrm{LF}^{\Box_{\preceq_o}}$, continued.

$$\text{Simple Types} \quad \alpha, \beta \quad ::= \quad a \mid \alpha[a] \to \beta$$

$\boxed{A^- = \alpha}$

$$\begin{array}{rcl} a^- & = & a \\ (P\ N)^- & = & P^- \\ (\Pi u{::}A_2[a].\, A)^- & = & A_2^-[a] \to A^- \end{array}$$

Figure 9: Erasure to Simple Types

we should be justified in using hereditary substitution to replace a hypothesis with a well-typed canonical term, and we should be able to construct a well-typed canonical term from a hypothesis. These correspond to global soundness and completeness, respectively.

7.1 Properties of Hereditary Substitution

Before proving the substitution and identity principles, we need a few properties of hereditary substitution. Since our definition of hereditary substitution is isomorphic to hereditary substitution for LF, these properties are standard and should be unsurprising. Nonetheless, we state them and sketch their proofs for the sake of completeness.

First of all, hereditary substitution is a partial function on its inputs:

Lemma 1 (Functionality of Substitution)**.**

1. *If* $[\![N_0/u_0]\!]^{n}_{\alpha_0[a_0]} N = N_1$ *and* $[\![N_0/u_0]\!]^{n}_{\alpha_0[a_0]} N = N_2$*, then* $N_1 = N_2$*.*
2. *If* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R = R_1$ *and* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R = R_2$*, then* $R_1 = R_2$*.*
3. *If* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R = N_1 : \alpha_1$ *and* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R = N_2 : \alpha_2$*, then* $N_1 = N_2$ *and* $\alpha_1 = \alpha_2$*.*

and similarly for the remaining syntactic categories.

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

Next, erasure to simple types is invariant under substitution:

Lemma 2 (Invariance of Erasure under Substitution)**.**

1. *If* $[\![N_0/u_0]\!]^{p}_{\alpha_0[a_0]} P = P'$*, then* $P^- = (P')^-$*.*
2. *If* $[\![N_0/u_0]\!]^{a}_{\alpha_0[a_0]} A = A'$*, then* $A^- = (A')^-$*.*

Proof. By induction on the structures of P and A. Note that the first part does not need to appeal to the second part. □

Like ordinary substitution, the hereditary version leaves a term unchanged if the variable being substituted for does not appear in the term:

Lemma 3 (Vacuous Substitutions)**.**

1. *If* $u_0 \notin \mathrm{FV}(R)$*, then* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R = N : \alpha$ *is not derivable.*
2. *If* $u_0 \notin \mathrm{FV}(N)$*, then* $[\![N_0/u_0]\!]^{n}_{\alpha_0[a_0]} N = N$*.*
3. *If* $u_0 \notin \mathrm{FV}(R)$*, then* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R = R$*.*

and similarly for the other syntactic categories.

Proof. By induction on the structure of the object into which the substitution occurs. Note that the induction need not be mutual across all parts. For example, the first part can be proved independently of the others, and the second and third parts can be proved together with an appeal to the first part. □

Ordinary substitution also possesses a commutativity property:

$$[e_0/x_0][e_2/x_2]e_1 = [([e_0/x_0]e_2)/x_2][e_0/x_0]e_1$$

Hereditary substitution does as well, though the statement and proof are complicated by relational definition (since hereditary substitution is not total) and the numerous syntactic categories. We have tried to mimic the indices used in the statement for ordinary substitution.

Lemma 4 (Composition of Substitutions)**.**
Suppose $[\![N_0/u_0]\!]^n_{\alpha_0[a_0]} N_2 = N_2'$, $u_2 \neq u_0$, *and* $u_2 \notin N_0$. *Then:*

1. *If* $[\![N_0/u_0]\!]^n_{\alpha_0[a_0]} N_1 = N_{1,0}$ *and* $[\![N_2/u_2]\!]^n_{\alpha_2[a_2]} N_1 = N_{1,2}$, *then* $[\![N_2'/u_2]\!]^n_{\alpha_2[a_2]} N_{1,0} = N$ *and* $[\![N_0/u_0]\!]^n_{\alpha_0[a_0]} N_{1,2} = N$.
2. *If* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R_1 = R_{1,0}$ *and* $[\![N_2/u_2]\!]^{rr}_{\alpha_2[a_2]} R_1 = R_{1,2}$, *then* $[\![N_2'/u_2]\!]^{rr}_{\alpha_2[a_2]} R_{1,0} = R$ *and* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R_{1,2} = R$.
3. *If* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R_1 = R_{1,0}$ *and* $[\![N_2/u_2]\!]^{rn}_{\alpha_2[a_2]} R_1 = N_{1,2} : \beta$, *then* $[\![N_2'/u_2]\!]^{rn}_{\alpha_2[a_2]} R_{1,0} = N : \beta$ *and* $[\![N_0/u_0]\!]^n_{\alpha_0[a_0]} N_{1,2} = N$.
4. *If* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R_1 = N_{1,0} : \beta$ *and* $[\![N_2/u_2]\!]^{rr}_{\alpha_2[a_2]} R_1 = R_{1,2}$, *then* $[\![N_2'/u_2]\!]^n_{\alpha_2[a_2]} N_{1,0} = N$ *and* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R_{1,2} = N : \beta$.

Proof. By lexicographic induction on the unordered pair $\{\alpha_0, \alpha_2\}$ and on the first given substitution derivation. □

7.2 Properties of Context Restriction

Because LF$^{\Box_{\preceq^o}}$'s novel feature is a context restriction, it will be useful to characterize its properties, notably the interaction between hereditary substitution and context restriction.

First, we prove a few algebraic properties:

Lemma 5 (Distributivity of Context Restriction)**.**
For all contexts Δ *and* Δ', *we have* $(\Delta, \Delta')|^{\preceq^o}_{\widehat{a}} = \Delta|^{\preceq^o}_{\widehat{a}}, \Delta'|^{\preceq^o}_{\widehat{a}}$.

Proof. By straightforward induction on Δ'. □

Lemma 6 (Idempotence of Context Restriction)**.**
For all Δ*, if* $a_0 \preceq_o a$*, then* $\Delta|^{\preceq_o}_{a}|^{\preceq_o}_{a_0} = \Delta|^{\preceq_o}_{a_0}$ *and* $\Delta|^{\preceq_o}_{a_0}|^{\preceq_o}_{a} = \Delta|^{\preceq_o}_{a_0}$*.*

Proof. By straightforward induction on Δ. □

Since substitution on contexts does not affect the type family constants that index the validity judgments, it follows that context restriction commutes with substitution:

Lemma 7 (Restriction Commutes with Substitution)**.**
If $[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}\Delta = \Delta'$*, then* $[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}(\Delta|^{\preceq_o}_{a}) = \Delta'|^{\preceq_o}_{a}$*.*

Proof. By induction on the structure of Δ. We show one case for nonempty Δ.

Case: $\Delta = \Delta_1, u{::}A_1[a_1]$.

$\mathcal{D} :: [\![N_0/u_0]\!]^c_{\alpha_0[a_0]}(\Delta_1, u{::}A_1[a_1]) = \Delta'$ Given
$\Delta' = \Delta'_1, u{::}A'_1[a_1]$ where
$\mathcal{D}_1 :: [\![N_0/u_0]\!]^c_{\alpha_0[a_0]}\Delta_1 = \Delta'_1$ and $\mathcal{D}_2 :: [\![N_0/u_0]\!]^a_{\alpha_0[a_0]}A_1 = A'_1$ By inversion
$[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}(\Delta_1|^{\preceq_o}_{a}) = \Delta'_1|^{\preceq_o}_{a}$ By i.h. on $\mathcal{D}_1$

Subcase: $a_1 \preceq_o a$
$[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}(\Delta_1|^{\preceq_o}_{a}, u{::}A_1[a_1]) = \Delta'_1|^{\preceq_o}_{a}, u{::}A'_1[a_1]$ By SUBST_C_CONS rule
$(\Delta_1, u{::}A_1[a_1])|^{\preceq_o}_{a} = \Delta_1|^{\preceq_o}_{a}, u{::}A_1[a_1]$ By definition of $|^{\preceq_o}_{a}$
$(\Delta'_1, u{::}A'_1[a_1])|^{\preceq_o}_{a} = \Delta'_1|^{\preceq_o}_{a}, u{::}A'_1[a_1]$ By definition of $|^{\preceq_o}_{a}$
$[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}((\Delta_1, u{::}A_1[a_1])|^{\preceq_o}_{a}) = (\Delta'_1, u{::}A'_1[a_1])|^{\preceq_o}_{a}$ By equality

Subcase: $a_1 \not\preceq_o a$
$(\Delta_1, u{::}A_1[a_1])|^{\preceq_o}_{a} = \Delta_1|^{\preceq_o}_{a}$ By definition of $|^{\preceq_o}_{a}$
$(\Delta'_1, u{::}A'_1[a_1])|^{\preceq_o}_{a} = \Delta'_1|^{\preceq_o}_{a}$ By definition of $|^{\preceq_o}_{a}$
$[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}((\Delta_1, u{::}A_1[a_1])|^{\preceq_o}_{a}) = (\Delta'_1, u{::}A'_1[a_1])|^{\preceq_o}_{a}$ By equality

$[\![N_0/u_0]\!]^c_{\alpha_0[a_0]}(\Delta|^{\preceq_o}_{a}) = \Delta'|^{\preceq_o}_{a}$ By replacing equals with equals

□

As alluded to in the discussion of context well-formedness, the rules were designed so that context restriction preserves context well-formedness. This property will be necessary for the substitution principle on contexts.

Lemma 8 (Context Restriction Preserves Well-Formedness)**.**
If $\vdash_{\Sigma,\preceq_o} \Delta$ `vctx`*, then* $\vdash_{\Sigma,\preceq_o} \Delta|^{\preceq_o}_{a}$ `vctx` *for all* $a \in \operatorname{dom}\Sigma$*.*

Proof. By induction on the structure of Δ. We again show the case for nonempty Δ.

Case: $\Delta = \Delta', u_0{::}A_0[a_0]$

$\vdash \Delta$ `vctx`	Given
$\vdash \Delta'$ `vctx` and $\Delta'|^{\preceq_o}_{a_0} \vdash A_0$ type	By inversion
$\vdash \Delta'|^{\preceq_o}_{a}$ `vctx`	By i.h. on Δ'

Subcase: $a_0 \preceq_o a$

$\Delta'|^{\preceq_o}_{a}|^{\preceq_o}_{a_0} = \Delta'|^{\preceq_o}_{a_0}$	By idempotence of $|^{\preceq_o}$
$\Delta'|^{\preceq_o}_{a}|^{\preceq_o}_{a_0} \vdash A_0$ type	By equality
$\vdash \Delta'|^{\preceq_o}_{a}, u_0{::}A_0[a_0]$ `vctx`	By VCTX_CONS rule
$\vdash (\Delta', u_0{::}A_0[a_0])|^{\preceq_o}_{a}$ `vctx`	By definition of $|^{\preceq_o}_{a}$

Subcase: $a_0 \not\preceq_o a$

$\vdash \Delta'|^{\preceq_o}_{a}$ `vctx`	From above
$\vdash (\Delta', u_0{::}A_0[a_0])|^{\preceq_o}_{a}$ `vctx`	By definition of $|^{\preceq_o}_{a}$

□

7.3 Substitution Principle

Before discussing the substitution principle, we expect to have weakening for $\text{LF}^{\Box_{\preceq_o}}$:

Theorem 9 (Weakening)**.** *Let $\mathcal{J}$ be one of the formation judgments. If $\Delta, \Delta' \vdash \mathcal{J}$ and $u \notin \operatorname{dom}\Delta \cup \operatorname{dom}\Delta'$, then $\Delta, u{::}A[a], \Delta' \vdash \mathcal{J}$.*

Proof. By structural induction on the first premise. □

At this point, we would like to prove a substitution principle for $\text{LF}^{\Box_{\preceq_o}}$:

Theorem 10 (Substitution)**.**

1. *If $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ `vctx` and $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash C$ type and $\Delta_1|^{\preceq_o}_{a_0} \vdash N_0 \Leftarrow A_0$ and $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash N \Leftarrow C$, then $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ and $\vdash \Delta_1, \Delta_2'$ `vctx`, and $[\![N_0/u_0]\!]^a_{A_0^-[a_0]}C = C'$ and $\Delta_1, \Delta_2' \vdash C'$ type, and $[\![N_0/u_0]\!]^n_{A_0^-[a_0]}N = N'$ and $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$.*
2. *If $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ `vctx` and $\Delta_1|^{\preceq_o}_{a_0} \vdash N_0 \Leftarrow A_0$ and $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash R \Rightarrow C$, then $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ and $\vdash \Delta_1, \Delta_2'$ `vctx`, and $[\![N_0/u_0]\!]^a_{A_0^-[a_0]}C = C'$ and either:*

- $[\![N_0/u_0]\!]^{rr}_{A_0^-[a_0]} R = R'$ *and* $\Delta_1, \Delta_2' \vdash R' \Rightarrow C'$, *or*
- $[\![N_0/u_0]\!]^{rn}_{A_0^-[a_0]} R = N' : (C')^-$ *and* $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$.

and similarly for the remaining syntactic categories.

Attempt 1: Following the lead of other canonical forms presentations of LF frameworks [WCPW02, HL07, LP08], our first attempt was to prove a series of proto-substitution lemmas that do not assume the well-formedness of various components. For example, we first attempted to prove proto-substitution for terms:

Lemma 11 (Proto-substitution for Terms)**.**

1. *If* $\Delta_1|^{\preceq^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash N \Leftarrow C$ *and* $[\![N_0/u_0]\!]^c_{A_0^-[a_0]} \Delta_2 = \Delta_2'$ *and* $[\![N_0/u_0]\!]^a_{A_0^-[a_0]} C = C'$, *then* $[\![N_0/u_0]\!]^n_{A_0^-[a_0]} N = N'$ *and* $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$.
2. *If* $\Delta_1|^{\preceq^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash R \Rightarrow C$ *and* $[\![N_0/u_0]\!]^c_{A_0^-[a_0]} \Delta_2 = \Delta_2'$ *then* $[\![N_0/u_0]\!]^a_{A_0^-[a_0]} C = C'$ *and either:*
   - $[\![N_0/u_0]\!]^{rr}_{A_0^-[a_0]} R = R'$ *and* $\Delta_1, \Delta_2' \vdash R' \Rightarrow C'$, *or*
   - $[\![N_0/u_0]\!]^{rn}_{A_0^-[a_0]} R = N' : (C')^-$ *and* $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$.

Unfortunately, a standard proof by mutual lexicographic induction on the simple type A_0^- and then on the typing derivation which hypothesizes $u_0{::}A_0[a_0]$ fails. To see why, consider the case for the ΠE rule when $a_0 \not\preceq^o a$:

$$\frac{\begin{array}{ccc}\mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 \\ \Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash R_1 \Rightarrow \Pi u{::}A_2[a].\, A & \Delta_1|^{\preceq^o}_{a}, \Delta_2|^{\preceq^o}_{a} \vdash N_2 \Leftarrow A_2 & [\![N_2/u]\!]^a_{A_2^-[a]} A = C\end{array}}{\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash R_1\ N_2 \Rightarrow C}\ \Pi E$$

As u_0 is no longer in scope when checking N_2, we would like to argue that the vacuous substitutions $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}(\Delta_2|^{\preceq^o}_{a}) = \Delta_2|^{\preceq^o}_{a}$, $[\![N_0/u_0]\!]^n_{A_0^-[a_0]} N_2 = N_2$, and $[\![N_0/u_0]\!]^a_{A_0^-[a_0]} A_2 = A_2$ exist. This way, we would be able to argue by functionality of substitution that $\mathcal{D}_2$ is the derivation of

$$\Delta_1|^{\preceq^o}_{a}, \Delta_2'|^{\preceq^o}_{a} \vdash N_2' \Leftarrow A_2'$$

which we need to reapply the ΠE rule to obtain the required result for this case. However, given the available assumptions, it is not true that $u_0 \notin \mathrm{FV}(\Delta_2|^{\preceq^o}_{a})$, which would be needed for the vacuous substitution. It seems

that the only way to guarantee that $u_0 \notin \mathrm{FV}(\Delta_2|^{\prec^o}_{a})$ is if the underlying context is well-formed.

Once we assume that the context is well-formed, other rules necessitate further well-formedness assumptions, resulting in a cascade. For example, the ΠI rule requires that we know that the type against which we check a canonical term is well-formed since that type is added to the context. Thus, a simple proto-substitution approach does not appear to work.

Attempt 2: Our second attempt was to prove a substitution lemma in one large statement:

Lemma 12 (Substitution)**.**

1. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$*,*
   then $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$*.*
2. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash K$ kind
   and $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$
   and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$*,*
   then $[\![N_0/u_0]\!]^k_{A_0^-[a_0]}K = K'$ *and* $\Delta_1, \Delta_2' \vdash K'$ kind*.*
3. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash A$ type
   and $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$
   and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$*,*
   then $[\![N_0/u_0]\!]^a_{A_0^-[a_0]}A = A'$ *and* $\Delta_1, \Delta_2' \vdash A'$ type*.*
4. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash P \Rightarrow K$
   and $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$
   and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$*,*
   then $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash K$ kind
   and $[\![N_0/u_0]\!]^k_{A_0^-[a_0]}K = K'$ *and* $\Delta_1, \Delta_2' \vdash K'$ kind
   and $[\![N_0/u_0]\!]^p_{A_0^-[a_0]}P = P'$ *and* $\Delta_1, \Delta_2' \vdash P' \Rightarrow K'$*.*
5. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash N \Leftarrow C$
   and $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$
   and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$
   and $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash C$ type
   and $[\![N_0/u_0]\!]^a_{A_0^-[a_0]}C = C'$ *and* $\Delta_1, \Delta_2' \vdash C'$ type*,*
   then $[\![N_0/u_0]\!]^n_{A_0^-[a_0]}N = N'$ *and* $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$*.*
6. *If* $\Delta_1|^{\prec^o}_{a_0} \vdash N_0 \Leftarrow A_0$ *and* $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash R \Rightarrow C$
   and $\vdash \Delta_1, u_0{::}A_0[a_0], \Delta_2$ $\mathtt{vctx}$
   and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]}\Delta_2 = \Delta_2'$ *and* $\vdash \Delta_1, \Delta_2'$ $\mathtt{vctx}$*,*

then $\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash C$ type
and $[\![N_0/u_0]\!]^c_{A_0^-[a_0]} C = C'$ *and* $\Delta_1, \Delta_2' \vdash C'$ type
and either:

- $[\![N_0/u_0]\!]^{rr}_{A_0^-[a_0]} R = R'$ *and* $\Delta_1, \Delta_2' \vdash R' \Rightarrow C'$, *or*
- $[\![N_0/u_0]\!]^{rn}_{A_0^-[a_0]} R = N' : (C')^-$ *and* $\Delta_1, \Delta_2' \vdash N' \Leftarrow C'$.

The lexicographic termination metric attempted for proto-substitution will not suffice here because the various cases call each other on inputs that are larger. Therefore, we follow the approach of Reed for HLF [Ree09], and attempt a lexicographic induction on the simple type A_0^-, then an ordering on the six cases, and finally on the derivation that hypothesizes $u_0{::}A_0[a_0]$. Rather than guessing a particular ordering, we attempted to uncover a correct one by carrying out the cases and then analyzing termination.

Unfortunately, we were again unable to find a correct ordering of the cases. To see why, consider the case for the $\Pi_K E$ rule when $a_0 \preceq_o a$:

$$\frac{\begin{array}{ccc}\mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 \\ \Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash P_1 \Rightarrow \Pi u{::}A_2[a].\, K & \Delta_1|^{\preceq_o}_{a}, u_0{::}A_0[a_0], \Delta_2|^{\preceq_o}_{a} \vdash N_2 \Leftarrow A_2 & [\![N_2/u]\!]^k_{A_2^-[a]} K = L\end{array}}{\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash P_1\ N_2 \Rightarrow L}\ \Pi_K E$$

To substitute for u_0 in the subderivation $\mathcal{D}_2$, we need to make an inductive call to substitution into canonical terms. Due to the ΠI rule, which extends the context with a component of the type, the statement of substitution into canonical terms requires the type to be well-formed. (Otherwise, we would not be able to preserve context well-formedness.) Therefore, we must supply a derivation of

$$\Delta_1|^{\preceq_o}_{a}, u_0{::}A_0[a_0], \Delta_2|^{\preceq_o}_{a} \vdash A_2 \text{ type}$$

before making the inductive call on $\mathcal{D}_2$. We can have this derivation by inversion, if we can find a derivation of

$$\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash \Pi u{::}A_2[a].\, K \text{ kind}$$

Since it doesn't seem possible to reconstruct this from an assumption about the well-formedness of L, we choose to produce a well-formedness derivation for the kind in the inductive hypothesis for atomic type families. (Producing, rather than assuming, this derivation is also consistent with the intended modes of the synthesis judgment.)

However, this therefore demands that, as part of this case, we establish

$$\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash L \text{ kind}$$

from the derivation obtained by inversion on the result of the inductive call on $\mathcal{D}_1$:

$$\Delta_1, u_0{::}A_0[a_0], \Delta_2, u{::}A_2[a] \vdash K \text{ kind}$$

The natural way to do this is by making an inductive call to substitute N_2 in for u. As the derivation may be large, this requires that the inductive hypothesis for kinds is ordered smaller than atomic type families.

Unfortunately, the inductive hypothesis for atomic type families must be no larger than the inductive hypothesis for kinds. This is because the $\Rightarrow\Leftarrow$ rule makes an inductive call on an atomic type family and the $\Pi_K F$ rule makes an inductive call on a canonical type, both when the simple type A_0^- remains the same. In other words, the hypothesis for atomic type families can be no larger than that for canonical types, which can be no larger than that for kinds.

For this reason, we were not able to make this attempt go through.

Attempt 3: Finally, we tried to revisit proto-substitution and insert a condition weaker than context well-formedness. Namely, we tried assuming, for all type family constants b, that $a_0 \not\preceq_o b$ implies $u_0 \notin \mathrm{FV}(\Delta_2|_b^{\prec_o})$. This, too, failed because of the ΠI rule:

$$\frac{\begin{array}{c}\mathcal{D}_1 \\ \Delta_1, u_0{::}A_0[a_0], \Delta_2, u{::}A_2[a] \vdash N_1 \Leftarrow A\end{array}}{\Delta_1, u_0{::}A_0[a_0], \Delta_2 \vdash \lambda u.N_1 \Leftarrow \Pi u{::}A_2[a].\, A}\ \Pi I$$

To make the inductive call on $\mathcal{D}_1$, we need to verify that $u_0 \notin \mathrm{FV}(A_2)$ whenever $a_0 \not\preceq_o a$ and $a \preceq_o b$. But this seems to require well-formedness of A_2 since there is no reason to expect a type to contain or not contain an arbitrary free variable.

Summary: As a result, we were unable to prove a substitution theorem for $\mathrm{LF}^{\Box_{\preceq_o}}$. At the same time, we were unable to find a counterexample to the desired substitution theorem: only termination of the various attempts seems to fail. Nonetheless, the absence of a proof is quite disappointing as it fails to support confidence in the framework's foundations. In the future, we would like to continue to search for a correct proof.

7.4 Expansion and Identity

In addition to a substitution theorem, the framework $\text{LF}^{\Box \preceq_o}$ should also possess an identity property that expresses its global completeness: morally, we want to show that $u{::}A[a] \vdash u \Leftarrow A$. Unfortunately, as in other LF logical frameworks, this statement is not correct because the $\Rightarrow\Leftarrow$ typing rule restricts the transition from atomic to canonical terms to occur at base types.

To state an identity theorem, we follow the standard approach and introduce η-expansion, which converts atomic terms to their canonical counterparts. The conversion is driven by the ostensible simple type of the term:

$\boxed{\eta_\alpha(R) = N}$

$$\begin{array}{lcl} \eta_a(R) & = & R \\ \eta_{\alpha_2[a]\to\alpha}(R) & = & \lambda u.\, \eta_\alpha(R\ \eta_{\alpha_2}(u)) \end{array}$$

The identity theorem can then be stated as a particular instance of the property that η-expansion preserves typing:

$$\text{If } A^- = \alpha, \text{ then } \Delta, u{::}A[a] \vdash \eta_\alpha(u) \Leftarrow A.$$

In the proof of this η-expansion theorem, we will need a lemma demonstrating that η-expansion commutes with substitution. To establish this, we require an auxiliary judgment that approximates the simple type of an atomic term when its head is a variable whose simple type is known:

$$\frac{}{u_0{::}\alpha_0[a_0] \vdash u_0 : \alpha_0} \qquad \frac{u_0{::}\alpha_0[a_0] \vdash R : \alpha_2[a] \to \alpha}{u_0{::}\alpha_0[a_0] \vdash R\ N : \alpha}$$

Note that the simple type produced by hereditary substitution into an atomic term matches this approximation:

Lemma 13.
If $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R = N' : \alpha'$ *and* $u_0{::}\alpha_0[a_0] \vdash R : \alpha$, *then* $\alpha' = \alpha$.

Proof. By induction on the structure of R. □

The statement of commutativity of η-expansion with substitution relies on a variable-only notion of substitution, $[v/u]E$. Unlike substitution of a canonical term for a variable, variable-only substitution can proceed compositionally without generating β-redices. This is because a variable at the head of an atomic term is only ever replaced by another variable.

Lemma 14 (η-Expansion Commutes with Substitution)**.**

1. (a) *If* $[\![\eta_\alpha(v)/u]\!]^n_{\alpha[a]} N = N'$, *then* $[v/u]N = N'$.
   (b) *If* $[\![\eta_\alpha(v)/u]\!]^{rr}_{\alpha[a]} R = R'$, *then* $[v/u]R = R'$.
   (c) *If* $[\![\eta_\alpha(v)/u]\!]^{rn}_{\alpha[a]} R = N : \beta$, *then* $[v/u](\eta_\beta(R)) = N$.

2. *If* $[\![N_0/u_0]\!]^n_{\alpha_0[a_0]} \eta_\alpha(R) = N'$ *and*

   (a) $|R| \neq u_0$, *then* $[\![N_0/u_0]\!]^{rr}_{\alpha_0[a_0]} R = R'$ *and* $\eta_\alpha(R') = N'$.
   (b) $|R| = u_0$ *and* $u_0{::}\alpha_0[a_0] \vdash R : \alpha$, *then* $[\![N_0/u_0]\!]^{rn}_{\alpha_0[a_0]} R = N' : \alpha$.

and similarly for the other syntactic categories.

Proof. By lexicographic induction on the structure of α and the given substitution derivation. □

Now we are finally ready to prove that η-expansion preserves typing. The development of η-expansion has thus far been completely standard, following very closely that of other LF logical frameworks (e.g., LFR [LP08]). Even here, the statement of and induction metric for the η-expansion theorem are standard. Nonetheless, due to the context restriction present in the ΠE typing rule, we must take special care to verify that the theorem holds.

Theorem 15 (η-Expansion)**.**
If $\Delta \vdash R \Rightarrow A$ *and* $A^- = \alpha$, *then* $\Delta \vdash \eta_\alpha(R) \Leftarrow A$.

Proof. By induction on the structure of α. We show the case for $\rightarrow$ simple types.

Case: $\alpha_2[a] \rightarrow \alpha$

$\Delta \vdash R \Rightarrow \Pi u{::}A_2[a].\, A$	Given
$\Delta, v{::}A_2[a] \vdash R \Rightarrow \Pi u{::}A_2[a].\, A$	By weakening
$\Delta|^{\preceq_o}_a, v{::}A_2[a] \vdash v \Rightarrow A_2$	By hypothesis rule
$\Delta|^{\preceq_o}_a, v{::}A_2[a] \vdash \eta_{\alpha_2}(v) \Leftarrow A_2$	By i.h. on α_2
$(\Delta, v{::}A_2[a])|^{\preceq_o}_a \vdash \eta_{\alpha_2}(v) \Leftarrow A_2$	By defn of $|^{\preceq_o}_a$ since $a \preceq_o a$
$\Delta, v{::}A_2[a] \vdash R\ \eta_{\alpha_2}(v) \Rightarrow [\![\eta_{\alpha_2}(v)/u]\!]^a_{\alpha_2[a]} A$	By ΠE rule
$\Delta, v{::}A_2[a] \vdash R\ \eta_{\alpha_2}(v) \Rightarrow [v/u]A$	By commutativity of subst. with η
$\Delta, u{::}A_2[a] \vdash \eta_\alpha(R\ \eta_{\alpha_2}(u)) \Leftarrow A$	By α-equivalence and i.h. on α
$\Delta \vdash \lambda u.\, \eta_\alpha(R\ \eta_{\alpha_2}(u)) \Leftarrow \Pi u{::}A_2[a].\, A$	By ΠI rule

□

Identity then follows immediately, since $\Delta, u{::}A[a] \vdash u \Rightarrow A$ is derivable by the hypothesis rule:

Corollary 16 (Identity)**.**
If $A^- = \alpha$, *then* $\Delta, u{::}A[a] \vdash \eta_\alpha(u) \Leftarrow A$.

7.5 Relationship to LF$^{\preceq_o}$

In Section 5, we sketched a framework, named LF$^{\preceq_o}$, which did not include meta-validity judgments and therefore could not provide the JS4 validity substitution principle for free. We claimed that LF$^{\Box_{\preceq_o}}$ would subsume LF$^{\preceq_o}$. We now argue why this is the case, though we do not present a formal proof since that would require a full description of LF$^{\preceq_o}$.

Recall that the ΠE rules for the two frameworks are:

LF$^{\preceq_o}$ rule:
$$\frac{\Gamma \vdash R \Rightarrow \Pi x{:}A_2.\, A \quad \Gamma|^{\preceq_o}_{A_2} \vdash N \Leftarrow A_2 \quad [N/x]^a_{A_2^-} A = A'}{\Gamma \vdash R\ N \Leftarrow A'}\ \Pi E$$

LF$^{\Box_{\preceq_o}}$ rule:
$$\frac{\Delta \vdash R \Rightarrow \Pi u{::}A_2[a].\, A \quad \Delta|^{\preceq_o}_{a} \vdash N \Leftarrow A_2 \quad [\![N/u]\!]^a_{A_2^-[a]} A = A'}{\Delta \vdash R\ N \Leftarrow A'}\ \Pi E$$

Suppose that we introduce a function $|A_2|$, which gives the type family constant at the head of a type:

$$\begin{array}{rcl} |a| & = & a \\ |P\ N| & = & |P| \\ |\Pi x{:}A_2.\, A| & = & |A| \end{array}$$

If we additionally impose the requirement that $\Pi u{::}A_2[a].\, A$ is only valid syntax in LF$^{\Box_{\preceq_o}}$ when $a = |A_2|$, then the two frameworks collapse. The index on a validity judgment becomes unnecessary since it can be extracted from that hypothesis's type. Moreover, under this requirement, the frameworks' context restrictions will behave the same way. That is, $\Gamma|^{\preceq_o}_{A_2}$ and $\Delta|^{\preceq_o}_{a}$ would both remove those hypotheses with head type family equal to $|A_2|$.

For this reason, we contend that LF$^{\Box_{\preceq_o}}$ is strictly more expressive than LF$^{\preceq_o}$, justifying our focus on LF$^{\Box_{\preceq_o}}$'s metatheory at the expense of LF$^{\preceq_o}$'s metatheory.

8 Adequacy of Encoding for JS4

At this point, we would like to consider an encoding of JS4 in LF$^{\Box_{\preceq_o}}$ and sketch a proof of its adequacy. The signature is given in Figure 10.

$$\texttt{o} : \mathsf{type}.$$

$$\texttt{imp} : \texttt{o}[\texttt{o}] \to \texttt{o}[\texttt{o}] \to \texttt{o}.$$
$$\texttt{box} : \texttt{o}[\texttt{o}] \to \texttt{o}.$$

$$\texttt{true} : \texttt{o}[\texttt{o}] \to \mathsf{type}.$$
$$\texttt{lvalid} : \texttt{o}[\texttt{o}] \to \mathsf{type}.$$

$$\texttt{o} \preceq_o \texttt{true}. \quad \texttt{lvalid} \preceq_o \texttt{true}.$$
$$\texttt{o} \preceq_o \texttt{lvalid}. \quad \texttt{true} \not\preceq_o \texttt{lvalid}.$$

$$\texttt{impI} : \Pi A{::}\texttt{o}[\texttt{o}].\, \Pi B{::}\texttt{o}[\texttt{o}].\\ ((\texttt{true}\ A)[\texttt{true}] \to \texttt{true}\ B)[\texttt{true}] \to \texttt{true}\ (\texttt{imp}\ A\ B).$$
$$\texttt{impE} : \Pi A{::}\texttt{o}[\texttt{o}].\, \Pi B{::}\texttt{o}[\texttt{o}].\\ (\texttt{true}\ (\texttt{imp}\ A\ B))[\texttt{true}] \to (\texttt{true}\ A)[\texttt{true}] \to \texttt{true}\ B.$$
$$\texttt{boxI} : \Pi A{::}\texttt{o}[\texttt{o}].\\ (\texttt{true}\ A)[\texttt{lvalid}] \to \texttt{true}\ (\texttt{box}\ A).$$
$$\texttt{boxE} : \Pi A{::}\texttt{o}[\texttt{o}].\, \Pi C{::}\texttt{o}[\texttt{o}].\\ (\texttt{true}\ (\texttt{box}\ A))[\texttt{true}] \to ((\texttt{true}\ A)[\texttt{lvalid}] \to \texttt{true}\ C)[\texttt{true}] \to \texttt{true}\ C.$$

Figure 10: Signature for an encoding of JS4.

$$\frac{A\ \mathit{prop} \gg N_A \Leftarrow \mathtt{o} \quad B\ \mathit{prop} \gg N_B \Leftarrow \mathtt{o}}{A \supset B\ \mathit{prop} \gg \mathtt{imp}\ N_A\ N_B \Leftarrow \mathtt{o}}\ \text{ENC\_O\_IMP}$$

$$\frac{A\ \mathit{prop} \gg N_A \Leftarrow \mathtt{o}}{\Box A\ \mathit{prop} \gg \mathtt{box}\ N_A \Leftarrow \mathtt{o}}\ \text{ENC\_O\_BOX}$$

Figure 11: Encoding of JS4 Propositions

In particular, note the absence of an explicit coercion from validity to truth; this is now captured by the $\text{LF}^{\Box\preceq o}$ hypothesis rule since the validity assumptions are encoded as $(\mathtt{true}\ A)[\mathtt{lvalid}]$. Also, note that the `boxI` constant handles the context-clearing effect directly by demanding a term of type $\mathtt{true}\ A$ that may contain only those open terms that are open-subordinate to `lvalid`.

In Figure 11, we present an encoding relation between JS4 propositions and $\text{LF}^{\Box\preceq o}$ terms of type `o`. For simplicity, we ignore propositional parameters, instead assuming that there exists an adequate encoding of at least one propositional constant. This allows us to focus on only the cases for implication and necessity. This encoding relation is a bijection:

Theorem 17 (Adequacy for Propositions)**.**

1. *If* $A\ \mathit{prop} \gg N \Leftarrow \mathtt{o}$*, then* $A\ \mathit{prop}$ *and* $\cdot \vdash N \Leftarrow \mathtt{o}$*.*
2. *If* $A\ \mathit{prop}$*, then there exists a unique term* N *such that* $A\ \mathit{prop} \gg N \Leftarrow \mathtt{o}$*.*
3. *If* $\cdot \vdash N \Leftarrow \mathtt{o}$*, then there exists a unique proposition* A *such that* $A\ \mathit{prop} \gg N \Leftarrow \mathtt{o}$*.*

Proof. By independent inductions on the structure of the given derivation. The third part requires a straightforward lemma for inversion of canonical terms of type `o`. □

Figure 12 presents an encoding relation for contexts. Assuming that fresh variables are used in this encoding, it is adequate:

Theorem 18 (Adqeuacy for Contexts)**.**

1. *If* $\delta; \gamma \gg \Delta\ \mathtt{vctx}$*, then* δ vctx *and* γ ctx *and* $\vdash \Delta\ \mathtt{vctx}$*.*
2. *If* δ vctx *and* γ ctx*, then there exists a unique context* Δ *(modulo* α*-equivalence and exchange) such that* $\delta; \gamma \gg \Delta\ \mathtt{vctx}$*.*
3. *If* $\vdash \Delta\ \mathtt{vctx}$*, then there exists a unique pair of contexts* $\delta; \gamma$ *(modulo* α*-equivalence and exchange) such that* $\delta; \gamma \gg \Delta\ \mathtt{vctx}$*.*

$$\frac{}{\cdot;\cdot \;\gg\; \cdot\ \texttt{vctx}}\ \text{ENC\_CTX\_NIL}$$

$$\frac{\delta;\gamma \;\gg\; \Delta\ \texttt{vctx} \quad A\ \mathit{prop} \;\gg\; N_A \Leftarrow \texttt{o}}{\delta;\gamma, x{:}A\ \mathit{true} \;\gg\; \Delta, u_x{::}(\texttt{true}\ N_A)[\texttt{true}]\ \texttt{vctx}}\ \text{ENC\_CTX\_CONS1}$$

$$\frac{\delta;\gamma \;\gg\; \Delta\ \texttt{vctx} \quad A\ \mathit{prop} \;\gg\; N_A \Leftarrow \texttt{o}}{\delta, u{::}A\ \mathit{valid};\gamma \;\gg\; \Delta, u_u{::}(\texttt{true}\ N_A)[\texttt{lvalid}]\ \texttt{vctx}}\ \text{ENC\_CTX\_CONS2}$$

Figure 12: Encoding of JS4 Contexts

Proof. By independent inductions on the given derivation(s). □

Finally, we can turn our attention toward proving adequacy for the encoding of JS4 derivations. The rules defining the encoding are given in Figure 13.

Theorem 19 (Adequacy for Derivations)**.**

1. *If* $\delta;\gamma \;\gg\; \Delta\ \texttt{vctx}$ *and* $\mathcal{D} :: \delta;\gamma \vdash A\ \mathit{true} \;\gg\; \Delta \vdash N \Leftarrow \texttt{true}\ N_A$*, then*
   - $A\ \mathit{prop} \;\gg\; N_A \Leftarrow \texttt{o}$,
   - $\mathcal{D}$ *derives* $\delta;\gamma \vdash A\ \mathit{true}$*, and*
   - $\Delta \vdash N \Leftarrow \texttt{true}\ N_A$.
2. *If* $\delta;\gamma \;\gg\; \Delta\ \texttt{vctx}$ *and* $\mathcal{D}$ *derives* $\delta;\gamma \vdash A\ \mathit{true}$*, then there exist unique* N *and* N_A *such that* $\Delta \vdash N \Leftarrow \texttt{true}\ N_A$.
3. *If* $\delta;\gamma \;\gg\; \Delta\ \texttt{vctx}$ *and* $\Delta \vdash N \Leftarrow \texttt{true}\ N_A$*, then there exist unique* A *and* $\mathcal{D}$ *such that* $\mathcal{D}$ *derives* $\delta;\gamma \vdash A\ \mathit{true}$.

Proof. By independent inductions on the second given derivations. We show the case for ENC_TRUE_BOXI in the first part.

Case: ENC_TRUE_BOXI

$\delta;\gamma \;\gg\; \Delta\ \texttt{vctx}$	Given
$\mathcal{D}_1 :: \delta;\cdot \vdash A\ \mathit{prop} \;\gg\; \Delta|^{\preceq_o}_{\texttt{lvalid}} \vdash N \Leftarrow \texttt{true}\ N_A$	Given
$\delta;\cdot \;\gg\; \Delta|^{\preceq_o}_{\texttt{lvalid}}$	By defn of encoding and $\preceq_o$
$A\ \mathit{prop} \;\gg\; N_A \Leftarrow \texttt{o}$ and $\mathcal{D}_1$ derives $\delta;\cdot \vdash A\ \mathit{true}$ and $\Delta|^{\preceq_o}_{\texttt{lvalid}} \vdash N \Leftarrow \texttt{true}\ N_A$	By i.h.(1)

$$\frac{}{\delta;\gamma,x{:}A\ \mathit{true},\gamma' \vdash A\ \mathit{true}}\ x \quad \gg\ \Delta, u_x{::}(\mathtt{true}\ N_A)[\mathtt{true}], \Delta' \vdash u_x \Leftarrow \mathtt{true}\ N_A\ \text{ENC\_TRUE\_HYP}$$

$$\frac{}{\delta, u{::}A\ \mathit{valid}, \delta';\gamma \vdash A\ \mathit{true}}\ u \quad \gg\ \Delta, u_u{::}(\mathtt{true}\ N_A)[\mathtt{valid}], \Delta' \vdash u_u \Leftarrow \mathtt{true}\ N_A\ \text{ENC\_TRUE\_VHYP}$$

$$\frac{\mathcal{D}_1 :: \delta;\gamma,x{:}A\ \mathit{true} \vdash B\ \mathit{true}}{\delta;\gamma \vdash A \supset B\ \mathit{true}}\ \supset I^x \quad \gg\ \Delta \vdash \mathtt{impI}\ N_A\ N_B\ (\lambda u_x.N) \Leftarrow \mathtt{true}\ (\mathtt{imp}\ N_A\ N_B)\ \text{ENC\_TRUE\_IMPI}$$

$$\frac{\mathcal{D}_1 :: \delta;\gamma \vdash A \supset B\ \mathit{true} \quad \gg\ \Delta \vdash N_1 \Leftarrow \mathtt{true}\ (\mathtt{imp}\ N_A\ N_B) \quad \mathcal{D}_2 :: \delta;\gamma \vdash A\ \mathit{true}}{\delta;\gamma \vdash B\ \mathit{true}}\ \supset E \quad \gg\ \Delta \vdash \mathtt{impE}\ N_A\ N_B\ N_1\ N_2 \Leftarrow \mathtt{true}\ N_B\ \text{ENC\_TRUE\_IMPE}$$

$$\frac{\mathcal{D}_1 :: \delta;\cdot \vdash A\ \mathit{true} \quad \gg\ \Delta|^{\preceq_o}_{\mathtt{valid}} \vdash N_1 \Leftarrow \mathtt{true}\ N_A}{\dfrac{\mathcal{D}_1 :: \delta;\cdot \vdash A\ \mathit{true}}{\delta;\gamma \vdash \Box A\ \mathit{true}}\ \Box I \quad \gg\ \Delta \vdash \mathtt{boxI}\ N_A\ N_1 \Leftarrow \mathtt{true}\ (\mathtt{box}\ N_A)}\ \text{ENC\_TRUE\_BOXI}$$

$$\frac{\mathcal{D}_1 :: \delta;\gamma \vdash \Box A\ \mathit{true} \quad \gg\ \Delta \vdash N_1 \Leftarrow \mathtt{true}\ (\mathtt{box}\ N_A) \quad \mathcal{D}_2 :: \delta, u{::}A\ \mathit{valid};\gamma \vdash C\ \mathit{true} \quad \gg\ \Delta, u_u{::}(\mathtt{true}\ N_A)[\mathtt{valid}] \vdash N_2 \Leftarrow \mathtt{true}\ N_C}{\delta;\gamma \vdash C\ \mathit{true}}\ \Box E^u \quad \gg\ \Delta \vdash \mathtt{boxE}\ N_A\ N_C\ N_1\ (\lambda u_u.N_2) \Leftarrow \mathtt{true}\ N_C\ \text{ENC\_TRUE\_BOXE}$$

Figure 13: Encoding of JS4 Derivations

$\Box A$ *prop* $\gg$ $\texttt{box}\ N_A \Leftarrow \texttt{o}$ By ENC_PROP_BOX rule
$\delta; \gamma \vdash \Box A$ *true* By $\Box I$ rule on $\mathcal{D}_1$

$\cdot \vdash N_A \Leftarrow \texttt{o}$ By adequacy of the encoding for propositions
$\Delta|^{\preceq_o}_{\texttt{o}} = \cdot$ By given $\preceq_o$ preorder
$\Delta \vdash \texttt{boxI}\ N_A\ N \Leftarrow \texttt{true}\ (\texttt{box}\ N_A)$ By $\text{LF}^{\Box_{\preceq_o}}$ typing rules

□

We would also like to know that the JS4 substitution principles are captured by substitution at the level of $\text{LF}^{\Box_{\preceq_o}}$ terms. This is stated as the following theorem:

Theorem 20 (Compositionality of the Encoding of Derivations)**.**

1. *If* $\delta; \gamma, x{:}A$ *true* $\gg$ $\Delta, u_x{::}(\texttt{true}\ N_A)[\texttt{true}]$
   and $\mathcal{D}_2 :: \delta; \gamma \vdash A$ *true* $\gg$ $\Delta \vdash N_2 \Leftarrow \texttt{true}\ N_A$
   and $\mathcal{D} :: \delta; \gamma, x{:}A$ *true* $\vdash C$ *true* $\gg$ $\Delta, u_x{::}(\texttt{true}\ N_A)[\texttt{true}] \vdash N \Leftarrow \texttt{true}\ N_C$
   and $[\mathcal{D}_2/x]\mathcal{D} = \mathcal{E}$ *and* $[\![N_2/u_x]\!]^n_{\texttt{true}[\texttt{true}]} N = N'$,
   then $\mathcal{E} :: \delta; \gamma \vdash C$ *true* $\gg$ $\Delta \vdash N' \Leftarrow \texttt{true}\ N_C$.
2. *If* $\delta, u{::}A$ *valid*$; \gamma \gg \Delta, u_u{::}(\texttt{true}\ N_A)[\texttt{lvalid}]$
   and $\mathcal{D}_2 :: \delta; \cdot \vdash A$ *true* $\gg$ $\Delta|^{\preceq_o}_{\texttt{lvalid}} \vdash N_2 \Leftarrow \texttt{true}\ N_A$
   and $\mathcal{D} :: \delta, u{::}A$ *valid*$; \gamma \vdash C$ *true* $\gg$ $\Delta, u_u{::}(\texttt{true}\ N_A)[\texttt{lvalid}] \vdash N \Leftarrow \texttt{true}\ N_C$
   and $[\![\mathcal{D}_2/u]\!]\mathcal{D} = \mathcal{E}$ *and* $[\![N_2/u_u]\!]^n_{\texttt{true}[\texttt{lvalid}]} N = N'$,
   then $\mathcal{E} :: \delta; \gamma \vdash C$ *true* $\gg$ $\Delta \vdash N' \Leftarrow \texttt{true}\ N_C$.

Assuming that a substitution theorem for $\text{LF}^{\Box_{\preceq_o}}$ can be proved, the compositionality of this encoding follows by induction on the encoding derivation.

9 Related Work

One method proposed in the literature for achieving a higher-order encoding of judgmental S4 [Cra09], based on a similar technique for encoding linear logic [Pfe94], is to use an auxiliary judgment on modal proof terms:

$$\texttt{local} : (\texttt{tm} \rightarrow \texttt{tm}) \rightarrow \mathsf{type}.$$

where the judgment $\texttt{local}\ (\lambda x.\, M_x)$ means that the variable x is not used within any validity proofs in M_x. Unfortunately, this places additional burden on the user in terms of spelling out the details of the `local` judgment

and proving `local`-specific metalemmata. In other words, the categorical nature of validity does not come for free. This technique thereby sacrifices some of the traditional advantages of LF and Twelf.

It is also possible to phrase modal logics in terms of explicit Kripke semantics [Sim94]. Since this formulation does not contain categorical judgments, it admits a straightforward encoding in LF [Mur08]. However, this does not uniformly resolve the case of general categorical judgments.

Also related is the Hybrid Logical Framework (HLF) [Ree09], developed as a linear metalogical framework, which labels LF assumptions with worlds relating them to the object-language context on which the corresponding object-language assumptions depend. It seems possible to encode categorical judgments in HLF. However, HLF requires new unification and coverage checking procedures, a requirement which we hope to avoid by staying as close to LF as possible.

Finally, the family of validity judgments used by $\text{LF}^{\Box_{\preceq_o}}$ seems technically similar to Nigam and Miller's subexponentials for linear logic [NM09]. Just as subexponentials are indexed by labels related by a preorder, the validity judgments of $\text{LF}^{\Box_{\preceq_o}}$ are indexed by type family constants and related by the open-terms subordination preorder. Differences include the absence of linearity and presence of dependent types in $\text{LF}^{\Box_{\preceq_o}}$ and the classical nature of Nigam and Miller's subexponentials.

10 Conclusion

In this project, we have proposed a novel refinement of subordination which deals specifically with open terms. We have demonstrated its connection to categorical judgments, using judgmental S4 modal logic (JS4) as our example. Based on this connection, we have proposed $\text{LF}^{\Box_{\preceq_o}}$, a logical framework that supports higher-order encodings of categorical judgments. We were able to prove identity and η-expansion theorems, but, disappointingly, could not prove (or disprove) a substitution theorem. Finally, we presented an adequate encoding of JS4 in $\text{LF}^{\Box_{\preceq_o}}$. We now conclude with a few suggestions for future work.

Future Work. The future work of primary importance is to pin down a proof of a substitution theorem. Without this, $\text{LF}^{\Box_{\preceq_o}}$ is of dubious foundations. Once this is complete, there are at least two notable avenues for future work.

First, it would be useful to implement $\mathrm{LF}^{\Box_{\preceq_o}}$ as an extension of Twelf. This way, we might gain practical benefits by being able to formalize categorical judgments within Twelf. We conjecture that type checking and reconstruction would be the primary components that would need to be modified.

Second, we would like to consider support for indexed categorical judgments, such as K knows judgments from judgmental logics of knowledge [GBB+06]. Because the open-terms subordination relation in $\mathrm{LF}^{\Box_{\preceq_o}}$ is parametric with respect to type indices, subordination on a family

$$\texttt{knows} : \texttt{principal} \to \texttt{o} \to \mathsf{type}.$$

would be too coarse to capture the property that K knows is categorical with respect to L knows when $L \neq K$. Perhaps using a refined notion of open-terms subordination that accounts for type indices would permit clean support for indexed categorical judgments.

References

[Cra09] Karl Crary. Higher-order representation of substructural logics. Unpublished. Available at http://www.cs.cmu.edu/~crary/papers/2009/substruct.pdf., July 2009.

[GBB+06] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of authorization and knowledge. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, pages 297–312, 2006.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, January 1993.

[HL07] Robert Harper and Daniel R. Licata. Mechanizing metatheory in a logical framework. *Journal of Functional Programming*, 17(4–5):613–673, July 2007.

[LP08] William Lovas and Frank Pfenning. A bidirectional refinement type system for LF. Technical Report CMU-CS-08-129, Department of Computer Science, Carnegie Mellon University, May 2008.

[Mur08] Tom Murphy VII. *Modal Types for Mobile Code*. PhD thesis, Carnegie Mellon University, 2008.

[NM09] Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. In *Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 129–140, 2009.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *ACM Transactions Computational Logic*, 9(3):23:1–23:49, June 2008.

[PD01] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001. Notes to an invited talk at the *Workshop on Intuitionistic Modal Logics and Applications* (IMLA'99), Trento, Italy, July 1999.

[Pfe94] Frank Pfenning. Structural cut elimination in linear logic. Technical Report CMU-CS-94-222, Carnegie Mellon University, December 1994.

[PS99] Frank Pfenning and Carsten Schürmann. System description: Twelf – a meta-logical framework for deductive systems. In Harald Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, LNAI 1632, pages 202–206, Trento, Italy, July 1999. Springer-Verlag.

[Ree09] Jason Reed. *A Hybrid Logical Framework*. PhD thesis, Carnegie Mellon University, August 2009.

[Sim94] Alex K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

[Vir99] Roberto Virga. *Higher-Order Rewriting with Dependent Types*. PhD thesis, Carnegie Mellon University, 1999.

[WCPW02] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Carnegie Mellon University, 2002. Revised May 2003.