# LFCS

**Laboratory for Foundations of Computer Science**
Department of Computer Science - University of Edinburgh

# An Equational Formulation of LF

by

Robert Harper

An Equational Formulation of LF

**LFCS Report Series**

**ECS-LFCS-88-67**
**(also published as CSR-280-88)**

**OCTOBER 1988**

**LFCS**
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

# An Equational Formulation of LF

Robert Harper

## 1 Introduction

The LF type system [HHP87] was designed as a presentation language for formal deductive systems to serve as the basis for a logic-independent computer-assisted proof development environment. The LF type system has three levels: the level of *objects*, the level of *types and families*, and the level of *kinds*. The types classify the objects, and the kinds classify the types and families. There is one primitive type constructor, the generalized product of a family of types indexed by a type, and two primitive kinds, the kind of types and the generalized product of a family of kinds indexed by a type. The "has type" ("has kind") relation is invariant under conversion in the type (kind) position. The type checking problem is decidable. (See [HHP87] for further details.)

For the intended applications of LF, there is no need to consider equational theories.[1] There are, however, other reasons to do so. One reason is that Martin-Löf has recently developed a closely-related type system, called the system of *logical types* [NPS88], to serve as the foundation for a version of his constructive set theory. There set theory (and other formal systems such as the Logical Theory of Constructions [Smi84, Acz83, MA88]) are formalized by presenting a set of constants together with a set of equational axioms. For the case of set theory, there are two constants, $\mathsf{Set} : \mathsf{Type}$, the type of sets, and $\mathsf{El} : \mathsf{Set} \to \mathsf{Type}$, the type of elements of a set. The formalization of set theory proceeds by adding constants and equational axioms. For example, to define the function space constructor, one would introduce the constants

$$\begin{array}{rcl} \mathsf{arrow} & : & \mathsf{Set} \to \mathsf{Set} \to \mathsf{Set} \\ \mathsf{lambda} & : & \Pi A{:}\mathsf{Set}.\Pi B{:}\mathsf{Set}.\Pi b : \mathsf{El}(A) \to \mathsf{El}(B).\mathsf{El}(\mathsf{arrow}(A,B)) \\ \mathsf{apply} & : & \Pi A{:}\mathsf{Set}.\Pi B{:}\mathsf{Set}.\mathsf{El}(\mathsf{arrow}(A,B)) \to \mathsf{El}(A) \to \mathsf{El}(B) \end{array}$$

with the equation

$$\begin{array}{c} \lambda A{:}\mathsf{Set}.\lambda B{:}\mathsf{Set}.\lambda b : \mathsf{El}(A) \to \mathsf{El}(B).\lambda a : \mathsf{El}(A).\mathsf{apply}(\mathsf{lambda}(A,B,b),a) \\ = \\ \lambda A{:}\mathsf{Set}.\lambda B{:}\mathsf{Set}.\lambda b : \mathsf{El}(A) \to \mathsf{El}(B).\lambda a : \mathsf{El}(A).b(a) \\ : \\ \Pi A{:}\mathsf{Set}.\Pi B{:}\mathsf{Set}.\Pi b : \mathsf{El}(A) \to \mathsf{El}(B).\Pi a : \mathsf{El}(A).\mathsf{El}(B(a)). \end{array}$$

[1] But there may be reason to study a theory of definitions [Gri88].

See the monograph of Nordström, Petersson, and Smith [NPS88] for a more detailed exposition of this approach.

Another reason to study an equational variant of LF is that it is, in a sense, a "minimal" system of dependent types. It is therefore a good basis for considering the extension of the model-theoretic techniques developed for typed $\lambda$-calculi [BM84, BMM87, Fri75, Hen50, Mar75] to dependent types. The purpose of this report is to outline the development for the case of LF, and to suggest directions for further research. The main lesson is that the definitions follow the same pattern, but there are considerable technical complications. (This is my excuse for providing few details.)

The development of the basic definitions and their properties follows closely the pattern of [BMM87]. The main differences are directly attributable to the presence of dependent types. In particular, we must "stage" the definition of a frame and of interpretations of terms in a frame to get around the fact that it is impossible to state the consistency requirements on constant interpretations and environments without a prior definition of interpretation. Furthermore, in order to take full advantage of equational axioms, we must intermix equational axioms with constant declarations. This leads to a generalization of LF's notion of signature, and undermines the usual distinction between "language" and "theory." This means, in particular, that some signatures will have no non-trivial models, just as some theories have none in the usual setting.

The paper is organized as follows. In Section 1 we define an equational variant of LF, and state some of the important properties of that system. In Section 2 we define the notion of an extensional model for LF, and sketch a proof of the soundness and completeness of the equational theory with respect to the class of extensional models. In Section 3 we suggest directions for further research.

I am grateful to John Mitchell for several helpful discussions about this work, and to Furio Honsell for calling my attention to the need to consider non-extensional models as well.

# 2 The Equational Theory of LF

## 2.1 Syntax

Let Var be a countably infinite set of *variables*, ranged over by $x$, $y$, and $z$, and let Const be a countably infinite set of *constants*, disjoint from the variables, ranged over by $c$ and $d$. The set of *pre-terms* is given by the following grammar:

$$\begin{array}{rcl} K & ::= & \mathsf{Type} \mid \Pi x{:}A.K \\ A & ::= & c \mid \Pi x{:}A_1.A_2 \mid \lambda x{:}A_1.A_2 \mid AM \\ M & ::= & c \mid x \mid \lambda x{:}A.M \mid M_1 M_2 \end{array}$$

These classes are called, respectively, the *kind expressions*, the *type family expressions*, and the *object expressions*. All are collectively called *terms* or *expressions*. Let $J$, $K$, and $L$ range over the kind expressions, $A$, $B$, $C$, and $D$ over the type family expressions, and $M$, $N$, $P$, and $Q$ over the object expressions. Let $X$ and $Y$ range over all expressions.

The set of free variables of an expression, $\mathrm{FV}(X)$, is defined in the usual way, treating $\lambda$ and $\Pi$ as binding operators that bind the declared variable in the second argument.

A *signature* is a finite sequence of *constant declarations* of the form $c{:}A$ or $c{:}K$, and *equational hypotheses* of the form $M{=}N{:}A$, such that no constant is declared more than once. Note that we only consider as axioms equations between object expressions, and not between types or families of types. The empty signature is written $\bullet$, and non-empty signatures are written as comma-separated lists of declarations and hypotheses. The set $\mathrm{dom}(\Sigma)$ is the set of constants declared in $\Sigma$. We write $\Sigma(c)$ when $c \in \mathrm{dom}(\Sigma)$ for the unique $A$ or $K$ such that $c{:}A$ or $c{:}K$ occurs in $\Sigma$.

A *context* is a finite sequence of *variable declarations* of the form $x{:}A$ such that no variable is declared more than once. The empty context is written $\bullet$, and non-empty contexts are written as comma-separated lists of declarations. The set $\mathrm{dom}(\Gamma)$ is the set of variables declared in $\Gamma$. When $x \in \mathrm{dom}(\Gamma)$, we write $\Gamma(x)$ for the unique $A$ such that $x{:}A$ occurs in $\Gamma$, and $\Gamma_x$ for the prefix of $\Gamma$ up to, but not including, the declaration of $x$.

## 2.2 Formation and Equality Rules

In this section we present the set of *formation* and *equality* rules for the equational variant of LF. It is characteristic of systems of dependent types that formation and equality must be defined simultaneously, for equations between objects can entail equations between types, and hence can lead to additional typings. Conversely, equations make sense only for well-formed terms.

The *formation judgements* are as follows:

| | |
|---|---|
| $\Sigma$ *valid* | $\Sigma$ *is a valid signature* |
| $\vdash_\Sigma \Gamma$ | $\Gamma$ *is a valid context* |
| $\Gamma \vdash_\Sigma K$ | $K$ *is a valid kind* |
| $\Gamma \vdash_\Sigma A : K$ | $A$ *is a family of kind* $K$ |
| $\Gamma \vdash_\Sigma M : A$ | $M$ *is an object of type* $A$ |

It is technically convenient to let $\tau$ range over the right-hand sides of the latter three formation judgements, writing $\Gamma \vdash_\Sigma \tau$ to stand for any of these three judgement forms. The free-variables and substitution operations are extended to $\tau$ in the obvious way.

The *equality judgements* are as follows:

| | |
|---|---|
| $\Gamma \vdash_\Sigma K = L$ | $K$ *and* $L$ *are equal kinds* |
| $\Gamma \vdash_\Sigma A = B : K$ | $A$ *and* $B$ *are equal families of kind* $K$ |
| $\Gamma \vdash_\Sigma M = N : A$ | $M$ *and* $N$ *are equal objects of type* $A$ |

We let $\varepsilon$ range over the right-hand sides of these judgements, and extend the substitution and free-variables operations to $\varepsilon$ in the obvious way.

The valid formation and equality judgements are inductively defined by the rules appearing in Tables 1 through 10.

## 2.3 Technical Properties

The following two propositions summarize some technical properties of the formation and equality rules. Although they are stated separately for the sake of clarity, they must in fact be proved simultaneously since the formation and equality relations are mutually dependent.

**Proposition 2.1**

1. *If* $\Gamma \vdash_\Sigma \tau$, *then* $\mathrm{FV}(\tau) \subseteq \mathrm{dom}(\Gamma)$.
2. *If* $\Gamma \vdash_\Sigma \tau$, $\vdash_\Sigma \Gamma'$, *and for all* $x \in \mathrm{FV}(\tau)$, $\Gamma_x \vdash_\Sigma \Gamma(x) = \Gamma'(x) : \mathsf{Type}$, *then* $\Gamma' \vdash_\Sigma \tau$.
3. *Signatures, contexts, and classifiers are well-formed:*
   (a) *If* $\vdash_\Sigma \Gamma$, *then* $\Sigma$ *valid*.
   (b) *If* $\Gamma \vdash_\Sigma K$, *then* $\vdash_\Sigma \Gamma$.
   (c) *If* $\Gamma \vdash_\Sigma A : K$, *then* $\Gamma \vdash_\Sigma K$.
   (d) *If* $\Gamma \vdash_\Sigma M : A$, *then* $\Gamma \vdash_\Sigma A : \mathsf{Type}$.
4. *Unicity of classifier:*
   (a) *If* $\Gamma \vdash_\Sigma A : K$ *and* $\Gamma \vdash_\Sigma A : L$, *then* $\Gamma \vdash_\Sigma K = L$.
   (b) *If* $\Gamma \vdash_\Sigma M : A$ *and* $\Gamma \vdash_\Sigma M : B$, *then* $\Gamma \vdash_\Sigma A = B : \mathsf{Type}$.
5. *Substitution: If* $\Gamma, x{:}A \vdash_\Sigma \tau$ *and* $\Gamma \vdash_\Sigma M : A$, *then* $\Gamma \vdash_\Sigma [M/x]\tau$.

Note that the second property given above entails both strengthening (eliminating non-occurring variables) and weakening (adding non-occurring variables) for contexts.

The following proposition summarizes some technical properties of the equality rules:

**Proposition 2.2**

1. *If* $\Gamma \vdash_\Sigma K = L$, $\vdash_\Sigma \Gamma'$, *and for all* $x \in \mathrm{FV}(K) \cup \mathrm{FV}(L)$, $\Gamma_x \vdash_\Sigma \Gamma(x) = \Gamma'(x) : \mathsf{Type}$, *then* $\Gamma' \vdash_\Sigma K = L$.
2. *Well-formedness of equands:*
   (a) *If* $\Gamma \vdash_\Sigma K = L$, *then* $\Gamma \vdash_\Sigma K$ *and* $\Gamma \vdash_\Sigma L$.
   (b) *If* $\Gamma \vdash_\Sigma A = B : K$, *then* $\Gamma \vdash_\Sigma A : K$ *and* $\Gamma \vdash_\Sigma B : K$.
   (c) *If* $\Gamma \vdash_\Sigma M = N : A$, *then* $\Gamma \vdash_\Sigma M : A$ *and* $\Gamma \vdash_\Sigma N : A$.
3. *Invertibility of type and kind equations:*
   (a) *If* $\Gamma \vdash_\Sigma \Pi x{:}A.K = \Pi x{:}B.L$, *then* $\Gamma \vdash_\Sigma A = B : \mathsf{Type}$ *and* $\Gamma, x{:}A \vdash_\Sigma K = L$.
   (b) *If* $\Gamma \vdash_\Sigma \Pi x{:}A.C = \Pi x{:}B.D : \mathsf{Type}$, *then* $\Gamma \vdash_\Sigma A = B : \mathsf{Type}$ *and* $\Gamma, x{:}A \vdash_\Sigma C = D : \mathsf{Type}$.
4. *Substitution. If* $\Gamma, x{:}A \vdash_\Sigma \varepsilon$ *and* $\Gamma \vdash_\Sigma M : A$, *then* $\Gamma \vdash_\Sigma [M/x]\varepsilon$.

# 3 Extensional Models of LF

In this section we shall need to work with partial functions. We adopt the following conventions for our informal work. For $e$ some mathematical expression, we write $e\downarrow$ to indicate that $e$ is defined (has a value). The Kleene equality relation, $e \simeq e'$, is defined to mean $e\downarrow \vee e'\downarrow \supset e = e'$. An expression of the form $e'$, *where* $x = e$ is defined only if $e$ is defined. We write $\varphi : X \rightharpoonup Y$ to indicate that $\varphi$ is a partial function carrying some subset of $X$ to $Y$.

In order to cope with the complications introduced by dependent types, we develop the notion of model for LF in stages. First, we define the basic notion of a *frame* which provides the overall structure of a model. Then we define the interpretation of terms into a frame as a partial function of the derivation of well-formedness for the term. Partial functions are an artifact of the need to stage the model definition; in the end, the meaning functions will all be total on the intended domain. The meaning functions are defined by induction on derivations as a technical device to cope with the type and kind equality formation rules. We then show that any two derivations for a given term in compatible contexts have the same meaning, and so we can correctly speak of the meaning of a term. We then classify frames according to the signatures that they satisfy (in an appropriate sense), and similarly classify environments according to the contexts that they satisfy. Finally, we define a model of a signature to be a frame in which every well-formed term has an interpretation, in every environment appropriate for the context.

**Definition 3.1** *An* extensional frame *is a structure*

$$\mathcal{A} = (\mathcal{K}, \mathcal{F}, \mathcal{O}),$$

*where* $\mathcal{K}$ *is a* kind frame, $\mathcal{F}$ *is a* type family frame, *and* $\mathcal{O}$ *is an* object frame, *defined simultaneously as follows.*

*A* kind frame *is a tuple*

$$\mathcal{K} = (|\mathcal{K}|, \mathrm{Type}^{\mathcal{K}}, \mathrm{Pi}^{\mathcal{K}})$$

*where*

- $|\mathcal{K}|$ *is a set (of* kinds*).*
- $\mathrm{Type}^{\mathcal{K}} \in |\mathcal{K}|$ *is a distinguished kind.*
- *For every* $a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$,
$$\mathrm{Pi}^{\mathcal{K}}_a : (|\mathcal{O}|_a \to |\mathcal{K}|) \rightharpoonup |\mathcal{K}|$$
*is a partial function.*

*A* type family frame *is a structure*

$$\mathcal{F} = (|\mathcal{F}|, \mathrm{Pi}^{\mathcal{F}}, \Phi^{\mathcal{F}}, \mathcal{I}^{\mathcal{F}})$$

*where*

- $|\mathcal{F}| = \langle\, |\mathcal{F}|_k \,\rangle_{k \in |\mathcal{K}|}$ *is a family of sets.*
- *For every* $a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$ *and* $f : |\mathcal{O}|_a \to |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$ *such that* $\mathrm{Pi}^{\mathcal{K}}(a, f)$ *is defined,*

$$\Phi^{\mathcal{F}}_{a,f} : |\mathcal{F}|_{\mathrm{Pi}^{\mathcal{K}}(a,f)} \rightarrowtail \prod_{x \in |\mathcal{O}|_a} |\mathcal{F}|_{f(x)}$$

  *is an injective function.*
- *For every* $a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$,

$$\mathrm{Pi}^{\mathcal{F}}_a : (|\mathcal{O}|_a \to |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}) \rightharpoonup |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$$

  *is a partial function.*
- $\mathcal{I}^{\mathcal{F}} : \mathrm{Const} \rightharpoonup \bigcup_{k \in |\mathcal{K}|} |\mathcal{F}|_k$ *is a partial function.*

*An* object frame *is a structure*

$$\mathcal{O} = (|\mathcal{O}|, \Phi^{\mathcal{O}}, \mathcal{I}^{\mathcal{O}})$$

*where*

- $|\mathcal{O}| = \langle\, |\mathcal{O}|_a \,\rangle_{a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}}$ *is a family of sets.*
- *For every* $a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$ *and* $f : |\mathcal{O}|_a \to |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}$ *such that* $\mathrm{Pi}^{\mathcal{F}}(a, f)$ *is defined,*

$$\Phi^{\mathcal{O}}_{a,f} : |\mathcal{O}|_{\mathrm{Pi}^{\mathcal{F}}(a,f)} \rightarrowtail \prod_{x \in |\mathcal{O}|_a} |\mathcal{O}|_{f(x)}$$

  *is an injective function.*
- $\mathcal{I}^{\mathcal{O}} : \mathrm{Const} \rightharpoonup \bigcup_{a \in |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}}} |\mathcal{O}|_a$. □

We adopt the following notational conventions:

$$\begin{array}{rcl} \mathcal{T} & = & |\mathcal{F}|_{\mathrm{Type}^{\mathcal{K}}} \\ |\mathcal{F}| & = & \bigcup_{k \in |\mathcal{K}|} |\mathcal{F}|_k \\ |\mathcal{O}| & = & \bigcup_{a \in \mathcal{T}} |\mathcal{O}|_a \\ |\mathcal{A}| & = & |\mathcal{K}| \cup |\mathcal{F}| \cup |\mathcal{O}| \end{array}$$

An *environment* for a frame $\mathcal{A}$ is a partial function $\eta : \mathrm{Var} \to |\mathcal{O}|$. The environment $\eta[x \leftarrow u]$ is the environment that agrees with $\eta$ on variables in its domain other than $x$, and that sends $x$ to $u$.

**Definition 3.2** *Let $\Sigma$ be a valid signature, and let $\mathcal{A}$ be an extensional frame. The interpretations of kind, type family, and object expressions into $\mathcal{A}$ are defined simultaneously by induction on the structure of derivations.*

$$
\begin{array}{rcl}
[\![\Gamma \vdash_\Sigma \mathsf{Type}]\!]^{\mathcal{K}}\eta & = & \mathrm{Type}^{\mathcal{K}} \\
[\![\Gamma \vdash_\Sigma \Pi x{:}A.K]\!]^{\mathcal{K}}\eta & \simeq & \mathrm{Pi}^{\mathcal{K}}(a,\varphi),\ \textit{where}\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\varphi(u) \simeq [\![\Gamma, x : A \vdash_\Sigma K]\!]^{\mathcal{K}}\eta[x \leftarrow u] \\
 & & \\
[\![\Gamma \vdash_\Sigma c : K]\!]^{\mathcal{F}}\eta & \simeq & \mathcal{I}^{\mathcal{F}}(c)\ \textit{if}\ \mathcal{I}^{\mathcal{F}}(c) \in |\mathcal{F}|_k,\ \textit{where}\ k = [\![\bullet \vdash_\Sigma K]\!]^{\mathcal{K}}\emptyset \\
[\![\Gamma \vdash_\Sigma \Pi x{:}A.B : \mathsf{Type}]\!]^{\mathcal{F}}\eta & \simeq & \mathrm{Pi}^{\mathcal{F}}(a,\varphi),\ \textit{where}\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\varphi(u) \simeq [\![\Gamma, x{:}A \vdash_\Sigma B : \mathsf{Type}]\!]^{\mathcal{F}}\eta[x \leftarrow u] \\
[\![\Gamma \vdash_\Sigma \lambda x{:}A.B : \Pi x{:}A.K]\!]^{\mathcal{F}}\eta & \simeq & {\Phi^{\mathcal{F}}_{a,\varphi}}^{-1}(\varphi)\ \textit{where}\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\varphi(u) \simeq [\![\Gamma, x{:}A \vdash_\Sigma B : K]\!]^{\mathcal{F}}\eta[x \leftarrow u] \\
[\![\Gamma \vdash_\Sigma BN : [N/x]K]\!]^{\mathcal{F}}\eta & \simeq & \Phi^{\mathcal{F}}_{a,\psi}(\varphi)(u)\ \textit{where}\ \varphi = [\![\Gamma \vdash_\Sigma B : \Pi x{:}A.K]\!]^{\mathcal{F}}\eta, \\
 & & u = [\![\Gamma \vdash_\Sigma N : A]\!]^{\mathcal{O}}\eta,\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\psi(u) \simeq [\![\Gamma, x{:}A \vdash_\Sigma K]\!]^{\mathcal{F}}\eta[x \leftarrow u] \\
[\![\Gamma \vdash_\Sigma A : K]\!]^{\mathcal{F}}\eta & \simeq & [\![\Gamma \vdash_\Sigma A : L]\!]^{\mathcal{F}}\eta\ (\textit{kind equality rule}) \\
 & & \\
[\![\Gamma \vdash_\Sigma c : A]\!]^{\mathcal{O}}\eta & \simeq & \mathcal{I}^{\mathcal{O}}(c)\ \textit{if}\ \mathcal{I}^{\mathcal{O}}(c) \in |\mathcal{O}|_a,\ \textit{where}\ a = [\![\bullet \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\emptyset \\
[\![\Gamma \vdash_\Sigma x : A]\!]^{\mathcal{O}}\eta & \simeq & \eta(x)\ \textit{if}\ \eta(x) \in |\mathcal{O}|_a,\ \textit{where}\ a = [\![\Gamma_x \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta \\
[\![\Gamma \vdash_\Sigma \lambda x{:}A.M : \Pi x{:}A.B]\!]^{\mathcal{O}}\eta & \simeq & {\Phi^{\mathcal{O}}_{a,\varphi}}^{-1}(\varphi),\ \textit{where}\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\varphi(u) \simeq [\![\Gamma, x{:}A \vdash_\Sigma M : B]\!]^{\mathcal{O}}\eta[x \leftarrow u] \\
[\![\Gamma \vdash_\Sigma MN : [N/x]B]\!]^{\mathcal{O}}\eta & \simeq & \Phi^{\mathcal{O}}_{a,\psi}(\varphi)(u),\ \textit{where}\ \varphi = [\![\Gamma \vdash_\Sigma M : \Pi x{:}A.B]\!]^{\mathcal{O}}\eta, \\
 & & u = [\![\Gamma \vdash_\Sigma N : A]\!]^{\mathcal{O}}\eta,\ a = [\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]^{\mathcal{F}}\eta,\ \textit{and} \\
 & & \forall u \in |\mathcal{O}|_a.\psi(u) \simeq [\![\Gamma, x{:}A \vdash_\Sigma B : \mathsf{Type}]\!]^{\mathcal{O}}\eta[x \leftarrow u] \\
[\![\Gamma \vdash_\Sigma M : A]\!]^{\mathcal{O}}\eta & \simeq & [\![\Gamma \vdash_\Sigma M : B]\!]^{\mathcal{O}}\eta\ (\textit{type equality rule})
\end{array}
$$

□

We drop the superscripts on the meaning functions when they are clear from context.

Given $\Sigma$, $\Gamma$, and $M$, it is easy to see that two derivations, one of $\Gamma \vdash_\Sigma M : A_1$ and the other of $\Gamma \vdash_\Sigma M : A_2$ can differ only in the use of the kind and type equality rules, all others being syntax-directed. Since meaning is preserved under applications of either of these rules (by definition of the interpretation functions), it is easy to see that we obtain the following result:

**Proposition 3.3**

1. $[\![\Gamma \vdash_\Sigma A : K]\!]\eta \simeq [\![\Gamma \vdash_\Sigma A : L]\!]\eta$ *whenever both judgements are derivable.*
2. $[\![\Gamma \vdash_\Sigma M : A]\!]\eta \simeq [\![\Gamma \vdash_\Sigma M : B]\!]\eta$ *whenever both judgements are derivable.*

Furthermore, if $\Gamma_1$ and $\Gamma_2$ agree, up to type equality, on all free variables occurring in $M$ and $A$, then $\Gamma_1 \vdash M : A$ is derivable iff $\Gamma_2 \vdash M : A$ is also derivable. Moreover, any two such derivations differ only on the uses of the kind and type equality rules. We therefore obtain the following result:

**Proposition 3.4** *If* $\Gamma \vdash_\Sigma \tau$, $\vdash_\Sigma \Gamma'$, *and for all* $x \in \mathrm{FV}(\tau)$, $\Gamma_x \vdash_\Sigma \Gamma(x) = \Gamma'(x) : \mathsf{Type}$, *then* $[\![\Gamma \vdash_\Sigma \tau]\!]\,\eta \simeq [\![\Gamma' \vdash_\Sigma \tau]\!]\,\eta$.

These two propositions may be summarized by saying that meaning is a function of the term and the types of its free variables, and is independent of the well-formedness derivation and of the types of non-occurring variables. Thus we may properly speak of the meaning of a term, given the types of its free variables.

There are several sources of undefinedness in the definition of the interpretation functions given above. First, a constant occurring in some derivation may not lie within the domain of $\mathcal{I}^{\mathcal{F}}$ or $\mathcal{I}^{\mathcal{O}}$ as required, or may lie within the domain, but be mapped to the "wrong" set. Second, the environment $\eta$ may either be undefined on, or assign an inappropriate value to, some variable occurring in the derivation. Third, some function $\varphi$ arising in the clauses for product types and kinds may not lie within the domain of $\mathrm{Pi}^{\mathcal{K}}$ or $\mathrm{Pi}^{\mathcal{F}}$, as required. Fourth, some function $\varphi$ arising in the interpretation of a $\lambda$-abstraction, either at the level of families or at the level of objects, may not lie within the range of $\Phi^{\mathcal{F}}$ or $\Phi^{\mathcal{O}}$, as required.

To deal with the first source of undefinedness, we classify frames according to the signatures with which they are consistent.[2]

**Definition 3.5** *Let* $\Sigma$ *be a valid signture. The property of being a* $\Sigma$-frame, *is defined by induction on the length of* $\Sigma$ *as follows:*

- *Every frame is a* $\bullet$*-frame.*
- *If* $\mathcal{A}$ *is a* $\Sigma$*-frame,* $[\![\bullet \vdash_\Sigma K]\!]\,\emptyset$ *exists and is equal to* $k$, *and* $\mathcal{I}^{\mathcal{F}}(c) \in |\mathcal{F}|_k$, *then* $\mathcal{A}$ *is a* $\Sigma, x{:}K$*-frame.*
- *If* $\mathcal{A}$ *is a* $\Sigma$*-frame,* $[\![\bullet \vdash_\Sigma A : \mathsf{Type}]\!]\,\emptyset$ *exists and is equal to* $a$, *and* $\mathcal{I}^{\mathcal{O}}(c) \in |\mathcal{O}|_a$, *then* $\mathcal{A}$ *is a* $\Sigma, x{:}A$*-frame.*
- *If* $\mathcal{A}$ *is a* $\Sigma$*-frame,* $[\![\bullet \vdash_\Sigma M : A]\!]\,\emptyset = [\![\bullet \vdash_\Sigma N : A]\!]\,\emptyset$, *then* $\mathcal{A}$ *is a* $\Sigma, M{=}N{:}A$*-frame.*

*We write* $\mathcal{A} \models \Sigma$ *to assert that* $\mathcal{A}$ *is a* $\Sigma$*-frame.*

Thus if $\mathcal{A}$ is a $\Sigma$-frame, the meaning of a derivation cannot fail to be defined on account of the non- or mis-interpretation of a constant.

To deal with the second source of undefinedness, we classify environments according the the contexts with which they are consistent.

[2]The following two definitions depend on some obvious structural properties of the formation rules such as the fact that a derivation of $\vdash_\Sigma \Gamma, x{:}A$ contains a derivation of $\Gamma \vdash_\Sigma A : \mathsf{Type}$ as a sub-derivation.

**Definition 3.6** *Let $\Sigma$ be a valid signature, let $\Gamma$ be such that $\vdash_\Sigma \Gamma$, and let $\mathcal{A}$ be a $\Sigma$-frame. The property of being a $\Gamma$-environment for $\mathcal{A}$ is defined by induction on the length of $\Gamma$ as follows:*

- *Every environment for $\mathcal{A}$ is a $\bullet$-environment for $\mathcal{A}$.*
- *If $\eta$ is a $\Gamma$-environment for $\mathcal{A}$, $[\![\Gamma_x \vdash_\Sigma A : \mathsf{Type}]\!]\,\emptyset$ exists and is equal to $a$, and $\eta(x) \in |\mathcal{O}|_a$, then $\eta$ is a $\Gamma, x{:}A$-environment for $\mathcal{A}$.*

*We write $\eta \models^{\mathcal{A}}_\Sigma \Gamma$ to assert that $\eta$ is a $\Gamma$-environment for $\Sigma$-frame $\mathcal{A}$.*

Thus if we restrict attention to environments appropriate for the given context, the interpretation of a term cannot fail to be defined on account of the non- or mis-interpertation of a variable.

We may now define our analogue of environment model [Mey82, BMM87].

**Definition 3.7** *Let $\Sigma$ be a valid signature, and let $\mathcal{A}$ be such that $\mathcal{A} \models \Sigma$. If for every derivable $\Gamma \vdash_\Sigma \tau$, and every $\eta$ such that $\eta \models^{\mathcal{A}}_\Sigma \Gamma$, all functions $\varphi$ arising in $[\![\Gamma \vdash_\Sigma \tau]\!]\,\eta$ lie within the domain of the appropriate* Pi *or $\Phi$ functions of $\mathcal{A}$, then $\mathcal{A}$ is said to be a* $\Sigma$-model.

It is clear that the definition of a $\Sigma$-model is somewhat unsatisfactory in that it expresses a "completeness" condition on the frame that is left unanalyzed. However, lacking a combinatory model definition, this is the best that we can do at present. See Section 4 below for further discussion.

**Proposition 3.8** *Let $\Sigma$ be a valid environment, let $\mathcal{A}$ be a $\Sigma$-model, and let $\eta$ be a $\Gamma$-context for $\mathcal{A}$. Then*

1. *All meanings exist: $[\![\Gamma \vdash_\Sigma \tau]\!]\,\eta\!\downarrow$.*
2. *Meanings are "well-typed:"*
   (a) *If $[\![\Gamma \vdash_\Sigma K]\!]\,\eta = k$ then $k \in |\mathcal{K}|$.*
   (b) *If $[\![\Gamma \vdash_\Sigma A : K]\!]\,\eta = a$ and $[\![\Gamma \vdash_\Sigma K]\!]\,\eta = k$, then $a \in |\mathcal{F}|_k$.*
   (c) *If $[\![\Gamma \vdash_\Sigma M : A]\!]\,\eta = u$ and $[\![\Gamma \vdash_\Sigma A : \mathsf{Type}]\!]\,\eta = a$, then $u \in |\mathcal{O}|_a$.*
3. *Interpretations respect equality:*
   (a) *If $\Gamma \vdash_\Sigma K = L$, then $[\![\Gamma \vdash_\Sigma K]\!]\,\eta = [\![\Gamma \vdash_\Sigma L]\!]\,\eta$.*
   (b) *If $\Gamma \vdash_\Sigma A = B : K$, then $[\![\Gamma \vdash_\Sigma A : K]\!]\,\eta = [\![\Gamma \vdash_\Sigma B : K]\!]\,\eta$.*
   (c) *If $\Gamma \vdash_\Sigma M = N : A$, then $[\![\Gamma \vdash_\Sigma M : A]\!]\,\eta = [\![\Gamma \vdash_\Sigma N : A]\!]\,\eta$.*
4. *Substitution: Suppose that $\Gamma \vdash_\Sigma M : A$. Then*

$$[\![\Gamma \vdash_\Sigma [M/x]\tau]\!]\,\eta = [\![\Gamma, x{:}A \vdash_\Sigma \tau]\!]\,\eta[x \leftarrow [\![\Gamma \vdash_\Sigma M : A]\!]\,\eta].$$

Validity of equations in LF models is defined as follows.

**Definition 3.9** *Define the relations* $\Gamma \models_\Sigma \varepsilon$ *by cases on* $\varepsilon$ *as follows:*

- *Suppose that* $\Gamma \vdash_\Sigma K$ *and* $\Gamma \vdash_\Sigma L$. *Define* $\Gamma \models_\Sigma K = L$ *iff for every* $\Sigma$*-frame* $\mathcal{A}$ *and every* $\Gamma$*-environment* $\eta$ *for* $\mathcal{A}$,

$$[\![\Gamma \vdash_\Sigma K]\!]\,\eta = [\![\Gamma \vdash_\Sigma L]\!]\,\eta.$$

- *Suppose that* $\Gamma \vdash_\Sigma A : K$ *and* $\Gamma \vdash_\Sigma B : K$. *Define* $\Gamma \models_\Sigma A = B : K$ *iff for every* $\Sigma$*-frame* $\mathcal{A}$ *and every* $\Gamma$*-environment* $\eta$ *for* $\mathcal{A}$,

$$[\![\Gamma \vdash_\Sigma A : K]\!]\,\eta = [\![\Gamma \vdash_\Sigma B : K]\!]\,\eta.$$

- *Suppose that* $\Gamma \vdash_\Sigma M : A$ *and* $\Gamma \vdash_\Sigma N : A$. *Define* $\Gamma \models_\Sigma M = N : A$ *iff for every* $\Sigma$*-frame* $\mathcal{A}$ *and every* $\Gamma$*-environment* $\eta$ *for* $\mathcal{A}$,

$$[\![\Gamma \vdash_\Sigma M : A]\!]\,\eta = [\![\Gamma \vdash_\Sigma N : A]\!]\,\eta.$$

The connection between the equational theory and the environment models is given by the following theorem.

**Theorem 3.10 (Soundness and Completeness)** $\Gamma \vdash_\Sigma \varepsilon$ *iff* $\Gamma \models_\Sigma \varepsilon$.

As usual, the proof breaks into two parts.

**Lemma 3.11 (Soundness)** *If* $\Gamma \vdash_\Sigma \varepsilon$, *then* $\Gamma \models_\Sigma \varepsilon$.

The soundness lemma is proved by a lengthy induction on derivations.

**Lemma 3.12 (Completeness)** *If* $\Gamma \models_\Sigma \varepsilon$, *then* $\Gamma \vdash_\Sigma \varepsilon$.

The proof of the completness theorem relies on the construction of a term model, for which we shall need to introduce a generalization of contexts. A *generalized context* is an infinite sequence of variable declarations such that no variable is declared more than once, and such that every prefix is a valid context in the usual sense. The formation relations are extended to generalized contexts by defining $\Delta \vdash_\Sigma \tau$ (where $\Delta$ is a generalized context) iff $\Delta_x \vdash_\Sigma \tau$ for some $x \in \mathrm{dom}(\Delta)$. (Note that $\Delta_x$ is a context in the ordinary sense.) A generalized context is *saturated* iff whenever $\Delta \vdash_\Sigma A : \mathsf{Type}$, then for infinitely many $x$ do we have $\Delta(x) = A$. It is easy to see that an arbitrary context $\Gamma$ may be extended to a saturated context.

The completeness proof goes as follows. Suppose $\Gamma \models_\Sigma \varepsilon$, and $\varepsilon$ is such that its equands are well-formed in the context $\Gamma$. We are to show $\Gamma \vdash_\Sigma \varepsilon$. Let $\Delta$ be a saturated extension of $\Gamma$. Define a frame whose sets consist of equivalence classes of terms well-formed in the context $\Delta$ under the equivalence relation of provable equality. An

environment for this frame maps variables to equivalence classes of terms. It is easy to see that the meaning of a term in this frame is the equivalence class of the substitution instance of the term induced by the environment. We obtain the desired result by considering the environment that maps each variable to its equivalence class, and observing that two terms have the same meaning in this environment iff they are provably equal in the context $\Delta$. But since these terms are sensible in the context $\Gamma$, and since $\Gamma$ and $\Delta$ agree, by construction of $\Delta$, on these variables, we may apply the strengthening rule for equality to conclude that they are provably equal in $\Gamma$, completing the proof.

To make this more precise, fix a signature $\Sigma$, and suppose that $\Gamma \models_\Sigma \varepsilon$ for some equation $\varepsilon$. Let $\Delta$ be a saturated extension of $\Gamma$. The relation $\equiv_{\text{kind}}$ is defined by

$$K \equiv_{\text{kind}} L \quad \textit{iff} \quad \Delta \vdash_\Sigma K = L.$$

The relations $A \equiv_K B$ and $M \equiv_A N$ are defined similarly. The latter two are obviously well-defined, for if $K \equiv_{\text{kind}} L$, then $A$ and $B$ are provably equal at kind $K$ iff they are provably equal at kind $L$, and similarly for equality of objects. We write $[K]$ for the equivalence class of $K$ under $\equiv_{\text{kind}}$, and similarly for $[A]$ and $[M]$ (their kind (type) being determined up to provably equality). The term frame $\mathcal{A}$ is defined to be $(\mathcal{K}, \mathcal{F}, \mathcal{O})$, where

$$\begin{aligned}
|\mathcal{K}| &= \{\, K \mid \Delta \vdash_\Sigma K \,\}/\equiv_{\text{kind}} \\
\text{Type}^{\mathcal{K}} &= [\mathsf{Type}] \\
\text{Pi}^{\mathcal{K}}([A], f) &= \Pi x{:}A.fx \textit{ where } f[M] = [fM]
\end{aligned}$$

$$\begin{aligned}
|\mathcal{F}|_{[K]} &= \{\, A \mid \Delta \vdash_\Sigma A : K \,\}/\equiv_K \\
\text{Pi}^{\mathcal{F}}([A], f) &= \Pi x{:}A.fx \textit{ where } f[M] = [fM] \\
\Phi^{\mathcal{F}}_{[A],f}([B])([M]) &= [BM] \\
\mathcal{I}^{\mathcal{F}}(c) &= [c]
\end{aligned}$$

$$\begin{aligned}
|\mathcal{O}|_{[A]} &= \{\, M \mid \Delta \vdash_\Sigma M : A \,\}/\equiv_A \\
\Phi^{\mathcal{O}}_{[A],f}([M])([N]) &= [MN] \\
\mathcal{I}^{\mathcal{O}}(c) &= [c]
\end{aligned}$$

It is easy to see that this is a well-defined extensional frame ($\eta$ plays an important role in showing extensionality.)

Suppose that $\eta \models^{\mathcal{A}}_\Sigma \Gamma$. Then $\eta(x) = [M]$ for some term $M$. Define $\eta^\sharp(X)$ to be the term resulting from the replacement of all free occurrences of $x \in \text{dom}(\eta)$ by some representative of $\eta(x)$. (It will not matter which we choose since provably equality is a congruence relation.) It is relatively easy to see that in $\mathcal{A}$, $[\![\Gamma \vdash_\Sigma K]\!]\,\eta = [\eta^\sharp(K)]$, and similarly for the other formation judgements. In particular, if we consider the environment $\eta_0$ defined by $\eta_0(x) = [x]$, then

$$[\![\Gamma \vdash_\Sigma K]\!]\,\eta_0 = [\![\Gamma \vdash_\Sigma L]\!]\,\eta_0 \quad \textit{iff} \quad \Delta \vdash_\Sigma K = L.$$

Now $\mathrm{FV}(K) \cup \mathrm{FV}(L) \subseteq \mathrm{dom}(\Gamma)$, and $\Delta$ and $\Gamma$ agree on $\mathrm{dom}(\Gamma)$, by the construction of $\Delta$. It follows that $\Gamma \vdash_\Sigma K = L$. This, together with similar arguments for the other cases, suffices to show completeness, for if an equation is valid in every $\Sigma$-model, it is valid in $\mathcal{A}$.

# 4 Directions for Further Work

In this paper we have presented an equational variant of the LF type theory, given a definition of an extensional model for this system, and outlined a proof of completeness for the theory with respect to this definition of model. Many more questions remain to be considered.

Meyer's notion of an environment model [Mey82] is a useful device for temporarily avoiding the question of the existence of sufficiently many functions in a frame to ensure that all terms may be interpreted. For many $\lambda$-calculi, this gap is filled by the definition of a combinatory model and the proof of combinatory completeness (see [Bar84, Mey82, BMM87], for example). Any serious consideration of the model theory of LF would have to address this question. The key problem seems to be to extend the notion of combinatory completeness to encompass product types and kinds, for a frame can fail to be a model if the required product types cannot be interpreted into it. The V3 type theory of Constable and Zlatin [CZ82] is a combinatory system of dependent types, but it relies on the existence of an infinite hierarchy of universes and an associated infinite set of combinators. It would be interesting to investigate whether it is necessary to incur such an infinite regress.

We have concentrated on extensional models in order to simplify the machinery. It seems clear that we may consider weakly extensional models by relaxing the injectivity condition on the $\Phi$'s, following the pattern of [BMM87]. The basic idea is to introduce a "selector" $\Psi$ that chooses a canonical "code" for a function of the appropriate semantic type in such a way that $\Psi$ is right inverse to $\Phi$.

In joint work with Andrzej Tarlecki (as yet unpublished) we have investigated "intensional" models of LF corresponding to the $\lambda$-algebras for the untyped $\lambda$-calculus [Bar84]. Here we expect $\xi$ to hold only as a rule of conversion (not equality). It is usual to define the $\lambda$-algebras using a combinatory structure, but lacking such we were forced to resort to the methods introduced by Hindley and Longo [HL80] (Barendregt's "syntactical $\lambda$-models" [Bar84]) to give an appropriate definition. Unfortunately such structures are rather unwieldy to manipulate. It would be interesting to reconsider such intensional interpretations in the light of a combinatory model definition for LF, if one can be found. (It seems that such structures would be the intended models for the aforementioned V3 type theory, but we have not verified this supposition.)

Set-theoretic interpretations may not always be appropriate. For example, Mitchell and Moggi have considered models of the simply-typed $\lambda$-calculus in functor categories in connection with empty types [MMMS87]. It would be interesting to examine the categorical structure of the LF type system. Hyland and Pitts [HP88] have developed

such structures for the Calculus of Constructions [CH85]; it seems likely that one can specialize their definition to the much simpler case of LF. The work of Seely [See84, See] and Cartmell [Car86] is also relevant here. It appears that a categorical study of the structure of LF models may suggest the definition of combinators for LF in a manner similar to that of Curien [Cur86] for the simply-typed $\lambda$-calculus.

# References


[Acz83] Peter Aczel. Frege structures revisited. In B. Nordström and J. Smith, editors, *Proceedings of the 1983 Marstrand Workshop*, 1983.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, revised edition, 1984.

[BM84] Kim Bruce and Albert R. Meyer. A completeness theorem for second-order polymorphic lambda calculus. In G. Kahn, D. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*, Lecture Notes in Computer Science, pages 131–144. Springer-Verlag, 1984.

[BMM87] Kim Bruce, Albert Meyer, and John C. Mitchell. The semantics of second-order lambda calculus. *Information and Computation*, 1987. To appear.

[Car86] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[CH85] Thierry Coquand and Gérard Huet. Constructions: A higher–order proof system for mechanizing mathematics. In B. Buchberger, editor, *EUROCAL '85: European Conference on Computer Algebra*, volume 203 of *Lecture Notes in Computer Science*, pages 151–184. Springer-Verlag, 1985.

[Cur86] P-L Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Research Notes in Theoretical Computer Science. Pitman/Wiley, 1986.

[CZ82] Robert L. Constable and Daniel R. Zlatin. Report on the type theory (V3) of the programming logic PL/CV3. In *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*. Springer-Verlag, 1982.

[Fri75] Harvey Friedman. Equality between functionals. In Rohit Parikh, editor, *Logic Colloquium, '75*, pages 22–37. North-Holland, 1975.

[Gri88] Timothy Griffin. Notational definition — a formal account. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 372–383, Edinburgh, July 1988.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.

[HHP87] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. In *Proceedings of the Symposium on Logic in Computer Science*, pages 194–204, Ithaca, New York, June 1987.

[HL80] J. R. Hindley and G. Longo. Lambda calculus models and extensionality. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 26:289–310, 1980.

[HP88] J. Martin E. Hyland and Andrew M. Pitts. The theory of constructions: Categorical semantics and topos-theoretic models. In *Proceedings of the Boulder Conference on Categories in Computer Science*, 1988. To appear.

[MA88] P. F. Mendler and P. Aczel. The notion of a framework and a framework for ltc. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 392–401, Edinburgh, July 1988.

[Mar75] Per Martin-Löf. About models for intuitionistic type theories and the notion of definitional equality. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, pages 81–109, Amsterdam, 1975. North-Holland.

[Mey82] Albert Meyer. What is a model of the lambda calculus? *Information and Control*, 52(1):87–122, 1982.

[MMMS87] J. C. Mitchell, A. R. Meyer, E. Moggi, and R. Statman. Empty types in polymorphic lambda calculus. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, pages 253–262, January 1987.

[NPS88] Bengt Nordström, Kent Petersson, and Jan Smith. *Programming in Martin-Löf's Type Theory.* University of Göteborg, Göteborg, Sweden, Midsummer 1988. Preprint.

[See] R. A. G. Seely. Categorical models for higher-order polymorphic lambda calculus. To appear in Journal of Symbolic Logic.

[See84] R. A. G. Seely. Locally cartesian closed categories and type theory. In *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 33–48. Cambridge Philosophical Society, 1984.

[Smi84] Jan Smith. An interpretation of Martin-Löf's type theory in a type-free theory of propositions. *Journal of Symbolic Logic*, 49(3):730–753, September 1984.

$$\bullet \ \mathit{valid} \tag{1}$$

$$\frac{\Sigma \ \mathit{valid} \quad \vdash_\Sigma A : \mathsf{Type} \quad c \notin \mathrm{dom}(\Sigma)}{\Sigma, c{:}A \ \mathit{valid}} \tag{2}$$

$$\frac{\Sigma \ \mathit{valid} \quad \vdash_\Sigma K \quad c \notin \mathrm{dom}(\Sigma)}{\Sigma, c{:}K \ \mathit{valid}} \tag{3}$$

$$\frac{\Sigma \ \mathit{valid} \quad \vdash_\Sigma M : A \quad \vdash_\Sigma N : A}{\Sigma, M{=}N{:}A \ \mathit{valid}} \tag{4}$$

Table 1: Signature Validity Rules

$$\frac{\Sigma \ \mathit{valid}}{\vdash_\Sigma \bullet} \tag{5}$$

$$\frac{\vdash_\Sigma \Gamma \quad \Gamma \vdash_\Sigma A : \mathsf{Type} \quad x \notin \mathrm{dom}(\Gamma)}{\vdash_\Sigma \Gamma, x{:}A} \tag{6}$$

Table 2: Context Validity Rules

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma \mathsf{Type}} \tag{7}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi x{:}A.K} \tag{8}$$

Table 3: Kinds

$$\frac{\vdash_\Sigma \Gamma \quad c{:}K \in \Sigma}{\Gamma \vdash_\Sigma c : K} \tag{9}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B : \mathsf{Type}}{\Gamma \vdash_\Sigma \Pi x{:}A.B : \mathsf{Type}} \tag{10}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda x{:}A.B : \Pi x{:}A.K} \tag{11}$$

$$\frac{\Gamma \vdash_\Sigma B : \Pi x{:}A.K \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma BN : [N/x]K} \tag{12}$$

$$\frac{\Gamma \vdash_\Sigma A : K \quad \Gamma \vdash_\Sigma K = L}{\Gamma \vdash_\Sigma A : L} \tag{13}$$

Table 4: Families of Types

$$\frac{\vdash_\Sigma \Gamma \quad c{:}A \in \Sigma}{\Gamma \vdash_\Sigma c : A} \tag{14}$$

$$\frac{\vdash_\Sigma \Gamma \quad x{:}A \in \Gamma}{\Gamma \vdash_\Sigma x : A} \tag{15}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x{:}A.M : \Pi x{:}A.B} \tag{16}$$

$$\frac{\Gamma \vdash_\Sigma M : \Pi x{:}A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : [N/x]B} \tag{17}$$

$$\frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma A = B : \mathsf{Type}}{\Gamma \vdash_\Sigma M : B} \tag{18}$$

Table 5: Objects

$$\frac{\Gamma \vdash_\Sigma K}{\Gamma \vdash_\Sigma K = K} \tag{19}$$

$$\frac{\Gamma \vdash_\Sigma A : K}{\Gamma \vdash_\Sigma A = A : K} \tag{20}$$

$$\frac{\Gamma \vdash_\Sigma M : A}{\Gamma \vdash_\Sigma M = M : A} \tag{21}$$

$$\frac{\Gamma \vdash_\Sigma K = L}{\Gamma \vdash_\Sigma L = K} \tag{22}$$

$$\frac{\Gamma \vdash_\Sigma A = B : K}{\Gamma \vdash_\Sigma B = A : K} \tag{23}$$

$$\frac{\Gamma \vdash_\Sigma M = N : A}{\Gamma \vdash_\Sigma N = M : A} \tag{24}$$

$$\frac{\Gamma \vdash_\Sigma J = K \quad \Gamma \vdash_\Sigma K = L}{\Gamma \vdash_\Sigma J = L} \tag{25}$$

$$\frac{\Gamma \vdash_\Sigma A = B : K \quad \Gamma \vdash_\Sigma B = C : K}{\Gamma \vdash_\Sigma A = C : K} \tag{26}$$

$$\frac{\Gamma \vdash_\Sigma M = N : A \quad \Gamma \vdash_\Sigma N = P : A}{\Gamma \vdash_\Sigma M = P : A} \tag{27}$$

Table 6: Equivalence Relation

$$\frac{\vdash_\Sigma \Gamma \quad M{=}N{:}A \in \Sigma}{\Gamma \vdash_\Sigma M = N : A} \tag{28}$$

$$\frac{\Gamma \vdash_\Sigma A = B : K \quad \Gamma \vdash_\Sigma K = L}{\Gamma \vdash_\Sigma A = B : L} \tag{29}$$

$$\frac{\Gamma \vdash_\Sigma M = N : A \quad \Gamma \vdash_\Sigma A = B : \mathsf{Type}}{\Gamma \vdash_\Sigma M = N : B} \tag{30}$$

$$\frac{\Gamma \vdash_\Sigma A = B : K \quad \vdash_\Sigma \Gamma' \quad \Gamma_x \vdash_\Sigma \Gamma(x) = \Gamma'(x) : \mathsf{Type}}{\Gamma' \vdash_\Sigma A = B : K} \tag{31}$$

$$(\forall x \in \mathrm{FV}(A) \cup \mathrm{FV}(B) \cup \mathrm{FV}(K))$$

$$\frac{\Gamma \vdash_\Sigma M = N : A \quad \vdash_\Sigma \Gamma' \quad \Gamma_x \vdash_\Sigma \Gamma(x) = \Gamma'(x) : \mathsf{Type}}{\Gamma' \vdash_\Sigma M = N : A} \tag{32}$$

$$(\forall x \in \mathrm{FV}(M) \cup \mathrm{FV}(N) \cup \mathrm{FV}(A))$$

Table 7: Structural Rules

$$\frac{\Gamma \vdash_\Sigma A = B : \mathsf{Type} \quad \Gamma, x{:}A \vdash_\Sigma K = L}{\Gamma \vdash_\Sigma \Pi x{:}A.K = \Pi x{:}B.L} \tag{33}$$

Table 8: Kind Equality

$$\frac{\Gamma \vdash_\Sigma A = B : \mathsf{Type} \quad \Gamma, x{:}A \vdash_\Sigma C = D : \mathsf{Type}}{\Gamma \vdash_\Sigma \Pi x{:}A.C = \Pi x{:}B.D : \mathsf{Type}} \tag{34}$$

$$\frac{\Gamma \vdash_\Sigma A = B : \mathsf{Type} \quad \Gamma, x{:}A \vdash_\Sigma C = D : K}{\Gamma \vdash_\Sigma \lambda x{:}A.C = \lambda x{:}B.D : \Pi x{:}A.K} \tag{35}$$

$$\frac{\Gamma \vdash_\Sigma B = C : \Pi x{:}A.K \quad \Gamma \vdash_\Sigma M = N : A}{\Gamma \vdash_\Sigma BM = CN : [M/x]K} \tag{36}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B : K \quad \Gamma \vdash_\Sigma M : A}{\Gamma \vdash_\Sigma (\lambda x{:}A.B)M = [M/x]B : [M/x]K} \tag{37}$$

$$\frac{\Gamma \vdash_\Sigma B : \Pi x{:}A.K \quad x \notin \mathrm{FV}(B)}{\Gamma \vdash_\Sigma \lambda x{:}A.Bx = B : \Pi x{:}A.K} \tag{38}$$

Table 9: Family Equality

$$\frac{\Gamma \vdash_\Sigma A = B : \mathsf{Type} \quad \Gamma, x{:}A \vdash_\Sigma M = N : C}{\Gamma \vdash_\Sigma \lambda x{:}A.M = \lambda x{:}B.N : \Pi x{:}A.C} \tag{39}$$

$$\frac{\Gamma \vdash_\Sigma M = N : \Pi x{:}A.B \quad \Gamma \vdash_\Sigma P = Q : A}{\Gamma \vdash_\Sigma MP = NQ : [P/x]B} \tag{40}$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma M : B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma (\lambda x{:}A.M)N = [N/x]M : [N/x]B} \tag{41}$$

$$\frac{\Gamma \vdash_\Sigma M : \Pi x{:}A.B \quad x \notin \mathrm{FV}(M)}{\Gamma \vdash_\Sigma \lambda x{:}A.Mx = M : \Pi x{:}A.B} \tag{42}$$

Table 10: Object Equality