Lambda-Free Logical Frameworks

Robin Adams

April 14, 2008

Abstract

We present the definition of the logical framework TF, the Type Framework. TF is a lambda-free logical framework; that is, it does not include lambda-abstraction or product kinds. We give formal proofs of several results in the metatheory of TF, and show how it can be conservatively embedded in the logical framework LF: its judgements can be seen as the judgements of LF that are in beta-normal, eta-long normal form. We show how several properties, such as adequacy theorems for object theories and the injectivity of constants, can be proven more easily in TF, and then ‘lifted’ to LF.

1 Introduction

A logical framework is a type theory intended as the meta-language for the specification of other formal systems, which may themselves be type theories, or could be other systems of logic. Traditionally, logical frameworks are based on a typed lambda calculus, and these are especially useful for specifying systems that involve binding of variables; variable binding is represented by lambda-abstraction in the framework, and substitution by application in the framework. However, this leads to a mismatch between the object theory and its representation in the framework. Each entity of the object theory is represented by more than one object in the framework — typically, beta-eta convertible objects represent the same entity of the object theory — and there are objects in the framework (such as partially applied meta-functions) that do not correspond to any entity of the object theory. It is therefore necessary to prove adequacy theorems establishing the relationship between an object theory and its representation in a logical framework; and these theorems are notoriously often difficult to prove.

It is possible to construct a logical framework that does not employ all the apparatus of the lambda calculus. We can construct logical frameworks that do not make use of abstraction and substitution, but instead involve only parametrisation and the instantiation of parameters; we shall call these lambda-free logical frameworks. They can be seen as frameworks that only use beta-normal, eta-long normal forms. Lambda-free framework provide a very faithful representation of an object theory — there is a one-to-one correspondence between the objects of the framework and the terms and types of the object theory.
Because of this, many results including adequacy theorems are easier to prove in a lambda-free framework.

It is often possible to embed a lambda-free framework \( L \) within a traditional framework \( F \); that is, to provide a translation from \( L \) into \( F \) such that the derivable judgements of \( L \) map onto exactly the derivable judgements of \( F \) that are in normal form. \( F \) can then be seen as a conservative extension of \( L \). Once this embedding has been established, we can ‘lift’ results from \( L \) to \( F \); that is, we can prove a result for \( L \), and then deduce that the corresponding result holds for \( F \) as a corollary.

There is a price to be paid for using a lambda-free frame: the earlier metatheoretic results are much more difficult to establish, as is the soundness of the embeddings discussed above. But this can be seen as a ‘one-time’ cost; once this price has been paid, it is comparatively easy to prove many results in the lambda-free frame, and then lift them to the traditional frames.

In this paper, we shall give the formal definition and several results from the metatheory of the lambda-free logical framework \( TF \), the Type Framework. In fact, we shall describe two versions of this framework; one Church-typed, which we call simply \( TF \), and one Curry-typed, which we call \( TF_k \). In Section 2 we present the formal definition of \( TF \). We prove several results from its metatheory in Section 3, with some of the more difficult proofs being relegated to Appendix A. We define \( TF_k \) in Section 4 and define mutually inverse translations between the two. We show in Section 5 how these two frames can be embedded in the logical framework \( LF \), a Church-typed version of Martin-Löf’s logical framework, and show how several properties of \( LF \) can be proven by lifting results from \( TF \).

1.1 Background

The term ‘lambda-free logical framework’ was first used to describe the framework \( PAL+ \), which uses parametrisation and definitions as its basic notions rather than lambda-abstraction. \( PAL+ \) does still have a mechanism for forming abstractions, however, namely parametric definition. We are using the phrase in a stricter sense, to describe a framework which has no abstraction mechanism at all.

The framework \( TF \) first appeared in an unpublished note by Aczel \[1\]. It was developed far beyond that presentation by the author in \[2\]; in particular, in that thesis the set of arities was introduced to organise the grammar, and the definition of instantiation was made explicit.

2 The Type Framework TF

We present our first example of a lambda-free framework, the Type Framework \( TF \). The framework \( TF \) includes nothing but what is essential for representing an object theory. In particular, it contains neither lambda-abstraction nor local
definition; its basic concepts are parameterisation, the instantiation of parameters, and the declaration of equations.

2.1 Grammar

2.1.1 Arities

We begin by introducing the set of arities, with which we shall organise the syntax of TF. They behave very similarly to the types of the simply typed lambda calculus.

The arities are defined inductively thus:

If $\alpha_1, \ldots, \alpha_n$ are arities, then $(\alpha_1, \ldots, \alpha_n)$ is an arity.

The base case of this definition is the case $n = 0$, yielding the arity $()$, which we shall write as $0$. The next arities that can be formed are

$$(0, \ldots, 0)$$

for positive $n$; we shall write this arity as $n$. The next arities that can be formed are $(n_1, \ldots, n_k)$, and so forth.

The intuition behind the arities is that an $(\alpha_1, \ldots, \alpha_n)$-ary function is a function that takes $n$ arguments — namely an $\alpha_1$-ary function, $\ldots$, and an $\alpha_n$-ary function — and returns an entity (term or type) of the object theory.

In particular, a 0-ary (or base) function is just an entity of the object theory; a 2-ary function is a binary operation on the entities of the object theory; and so forth.

We denote by $\alpha \hat{\beta}$ the concatenation of the two arities $\alpha$ and $\beta$:

$$(\alpha_1, \ldots, \alpha_m)(\beta_1, \ldots, \beta_n) \equiv (\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n).$$

We also ascribe an order to each arity as follows:

- The only 0th-order, or base, arity is $0$.
- If the highest order among the arities $\alpha_1, \ldots, \alpha_n$ is $k$, then $(\alpha_1, \ldots, \alpha_n)$ is a $k + 1$st-order arity.

In particular, the first-order arities are those of the form $n$ for positive $n$, and the second-order arities are those of the form $(n_1, \ldots, n_k)$ where at least one $n_i$ is positive.

2.2 Objects

The objects of TF are expressions intended to represent the terms and types of the object theory. They are built up from variables and constants, to each of which is assigned an arity. The constants shall be used for the type constructors and term constructors of the object theory. The variables shall be used as the variables of the object theory.

The set of objects is defined by the following inductive definition:
If $z$ is an $\alpha$-ary constant or variable, where

$$\alpha \equiv ((\alpha_{11}, \ldots, \alpha_{1r_1}), \ldots, (\alpha_{n1}, \ldots, \alpha_{nr_n})),$$

then

$$z[[x_{11}, \ldots, x_{1r_1}]M_1, \ldots, [x_{n1}, \ldots, x_{nr_n}]M_n] \quad (1)$$

is an object, where each $x_{ij}$ is an $\alpha_{ij}$-ary variable, and each $M_i$ an object. Each $x_{ij}$ is bound within the corresponding object $M_i$, and we identify objects up to $\alpha$-conversion.

The base case of this definition is that, if $z$ is a base variable or constant (that is, a 0-ary variable or constant), then $z[]$ is an object; we shall henceforth write this object as just $z$. Likewise, if $z$ is an $n$-ary variable or constant, then $z[[M_1, \ldots, [M_n]$ is an object for any objects $M_1, \ldots, M_n$; we shall write this object simply as $z[M_1, \ldots, M_n]$.

The subexpressions of the object (1) such as $[x_{1}, \ldots, x_{r}]M$ are not first-class entities of TF; they cannot occur except as arguments to some variable or constant $z$. Nevertheless, it shall be convenient to have some way of referring to these pieces of raw syntax. We shall therefore introduce the following terminology:

- An $(\alpha_1, \ldots, \alpha_n)$-ary variable sequence is a sequence of $n$ distinct variables $\langle x_1, \ldots, x_n \rangle$, where $x_i$ has arity $\alpha_i$.
- An $\alpha$-ary abstraction is an expression of the form $\langle \vec{x} \rangle M$, where $\vec{x}$ is an $\alpha$-ary variable sequence, and $M$ an object. We take each member of $\vec{x}$ to be bound within this abstraction, and identify abstractions up to $\alpha$-conversion.
- An $(\alpha_1, \ldots, \alpha_n)$-ary abstraction sequence is a sequence $\langle F_1, \ldots, F_n \rangle$, where $F_i$ is an $\alpha_i$-ary abstraction.

Thus, an object has the form $z[\vec{F}]$, where $z$ is an $\alpha$-ary variable or constant, and $\vec{F}$ an $\alpha$-ary abstraction sequence. We shall often write this object as just $z\vec{F}$.

We note that the only expressions that can occur as arguments to a symbol are abstractions. In the situations where we would naturally wish to write a variable or constant in an argument position, we instead write its $\eta$-long form.

**Definition 2.1 ($\eta$-long Form)** Given any $\alpha$-ary variable or constant $z$, the $\eta$-long form $z^\eta$ of $z$ is the $\alpha$-ary abstraction defined by recursion on $\alpha$ as follows:

If $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$, then

$$z^\eta \equiv [x_{1}, \ldots, x_{n}]z[x^\eta_{1}, \ldots, x^\eta_{n}] \quad ,$$

where each $x_i$ is an $\alpha_i$-ary variable. (By $\alpha$-conversion, it does not matter which variables we choose.)
2.3 Instantiation and Employment

We cannot use the familiar operation of substitution in TF. The result of substituting an abstraction $[\vec{y}]M$ for the variable $x$ in the object $xF$ is not an object of TF; rather, it would be a $\beta$-redex, which cannot be formed in TF.

Instead, we introduce an operation that we name instantiation. The operation of instantiating an abstraction $F$ for a variable $x$ can be thought of as substituting $F$ for $x$, then reducing to normal form (that is, $\beta$-normal, $\eta$-long form). However, we note that the definition does not use any notion of reduction.

**Definition 2.2 (Instantiation)** Given an $\alpha$-ary abstraction $F$, $\alpha$-ary variable $x$, and object $N$, the object $\{F/x\}N$, the result of instantiating $x$ with $F$ in $N$, is defined by recursion firstly on the arity $\alpha$, secondly on the object $N$, as follows:

$$\{F/x\}z[G_1, \ldots, G_n] \equiv z[\{F/x\}G_1, \ldots, \{F/x\}G_n] \quad (z \neq x)$$

If $F \equiv [t_1, \ldots, t_n]P$, then

$$\{F/x\}x[G_1, \ldots, G_n] \equiv \{\{F/x\}G_1/t_1\} \cdots \{\{F/x\}G_n/t_n\}P .$$

We assume here, through $\alpha$-conversion, that no $t_i$ occurs free in any $G_j$.

We shall also introduce a notational convention that shall play the role of abstraction: if $x$ is an $\alpha$-ary variable and $F$ a $\beta$-ary abstraction, then $[x]F$ is an $(\alpha)\hat{\beta}$-ary abstraction, defined by

$$[x][y_1, \ldots, y_n]M \equiv [x, y_1, \ldots, y_n]M .$$

Finally, we define an operation, which we shall call employment, to play the role usually taken by application. The result of employing $F$ on $G$, denoted $F \bullet G$, can be thought of as the normal form of the application $FG$. The definition is:

**Definition 2.3 (Employment)** Given an $(\alpha)\hat{\beta}$-ary abstraction $[x]F$ and an $\alpha$-ary abstraction $G$, the $\beta$-ary abstraction $F \bullet G$, the result of employing $[x]F$ on $G$, is defined by

$$(\{x\}F) \bullet G \equiv \{G/x\}F .$$

We have used our newly introduced notation $[x]M$ in this definition; written out in full, the above equation is

$$(\{x, y_1, \ldots, y_n\}M) \bullet G \equiv [y_1, \ldots, y_n]\{G/x\}M .$$

We shall abbreviate the repeated use of employment as follows: if $\vec{G}$ is the abstraction sequence $(G_1, \ldots, G_n)$, then $F \bullet \vec{G}$ abbreviates $F \bullet G_1 \bullet G_2 \bullet \cdots \bullet G_n$, that is,

$$((\cdots (F \bullet G_1) \bullet G_2) \bullet \cdots) \bullet G_n .$$

5
We have already noted the correspondence between the arities and the types of the simply-typed lambda calculus. In the same way, the fact that our definition of instantiation terminates corresponds to the weak normalisability of the simply-typed lambda calculus; specifically, to the fact that the strategy of innermost reduction always terminates.

2.4 Kinds

A base kind in TF is either the symbol Type, or has the form El(A) for some object A. The intention is that each type T of the object theory are represented by an object [T] of kind Type; the terms of type T are then represented by the objects of kind El([T]).

In addition to these, we introduce a set of $\alpha$-ary product kinds for every arity $\alpha$. These shall be used to give kinds to the variables and constants of higher arity. The definition is by recursion on $\alpha$:

An $(\alpha_1, \ldots, \alpha_n)$-ary product kind is an expression of the form

$$(x_1 : K_1, \ldots, x_n : K_n)T$$

where the $x_i$s are distinct variables, $x_i$ being of arity $\alpha_i$; each $K_i$ is an $\alpha_i$-ary product kind; and $T$ is a base kind.

We take each variable $x_i$ to be bound within $K_{i+1}, K_{i+2}, \ldots, K_n$ and $T$ in this product kind, and identify product kinds up to $\alpha$-conversion.

The intention is that an abstraction of the kind (2) is a function that takes $n$ arguments — namely $F_1$ of kind $K_1$, $F_2$ of kind $\{F_1/x_1\}K_2$, $\ldots$, and $F_n$ of kind $\{F_1/x_1, \ldots, F_{n-1}/x_{n-1}\}K_n$ — and returns an object of kind $\{F_1/x_1, \ldots, F_n/x_n\}T$.

We shall make use of a similar abuse of notation as we did for abstractions: if $K \equiv (x_1 : K_1, \ldots, x_n : K_n)T$, then we shall write $(y : J)K$ for $(y : J, x_1 : K_1, \ldots, x_n : K_n)T$.

Just as with abstractions, so the product kinds of non-zero arity are not considered first-class entities of TF; only the base kinds are. We shall however make use of the higher product kinds to give kinds to the variables and constants of higher arity. We shall even talk of an abstraction being of a product kind; however, this shall not be represented by a primitive judgement form of TF.

Contexts A context in TF is a sequence of the form:

$$x_1 : K_1, \ldots, x_n : K_n$$

where the $x_i$s are distinct variables, and each $x_i$ has the same arity as the corresponding kind $K_i$. If each $x_i$ has arity $\alpha_i$, we say the context has arity $(\alpha_1, \ldots, \alpha_n)$. (Thus, an $\alpha$-ary kind has the form $(\Delta)T$, where $\Delta$ is an $\alpha$-ary context and $T$ a base kind.) The variable sequence $\langle x_1, \ldots, x_n \rangle$ is called the domain of the context $\Delta$, $\text{dom} \Delta$. 
2.5 Judgement Forms

There are three primitive judgement forms in TF:

\[
\begin{align*}
\Gamma \text{ valid} \\
\Gamma \vdash M : T \\
\Gamma \vdash M = N : T
\end{align*}
\]

where \(\Gamma\) is a context, \(M\) and \(N\) are objects, and \(T\) is a base kind. These are intended to express that \(\Gamma\) is a valid context, that the object \(M\) has kind \(T\) under the context \(\Gamma\), and that the objects \(M\) and \(N\) are equal objects of kind \(T\) under \(\Gamma\), respectively.

We do not introduce primitive judgement forms to deal with the abstractions and product kinds of higher arity; rather, we introduce these as defined judgement forms. Every judgement involving these second-class entities is to be read as an abbreviation for a set of primitive judgements of the three forms given above. We shall always use the double turnstile \(\vdash\) to indicate such a defined judgement form.

For any base kind \(T\), the single judgement \(\Gamma \vdash T\) kind is defined as follows:

\[
\begin{align*}
\Gamma \vdash \text{Type} \text{ kind} & \equiv \Gamma \text{ valid} \\
\Gamma \vdash \text{El (}A\text{) kind} & \equiv \Gamma \vdash A : \text{Type}
\end{align*}
\]

For any \(\alpha\)-ary product kind \(K\), the judgement \(\Gamma \vdash K\) kind is defined by:

\[
\Gamma \vdash (\Delta)T \text{ kind} \equiv \Gamma, \Delta \vdash T \text{ kind}
\]

Equality of base kinds is defined by:

\[
\begin{align*}
\Gamma \vdash \text{Type} = \text{Type} & \equiv \Gamma \text{ valid} \\
\Gamma \vdash \text{El (}A\text{) = El (}B\text{) } & \equiv \Gamma \vdash A = B : \text{Type}
\end{align*}
\]

We leave \(\Gamma \vdash \text{Type} = \text{El (}B\text{)}\) and \(\Gamma \vdash \text{El (}A\text{) = Type}\) undefined.

Equality of product kinds and contexts is defined recursively by

\[
\begin{align*}
\Gamma \vdash (\Delta)T = (\Delta')T' & \equiv (\Gamma \vdash \Delta = \Delta') \cup \{\Gamma \vdash T = T'\} \\
\Gamma \vdash () = () & \equiv \{\text{valid}\} \\
\Gamma \vdash \Delta, x : K = \Delta', x : K' & \equiv (\Gamma \vdash \Delta = \Delta') \cup (\Gamma, \Delta \vdash K = K')
\end{align*}
\]

We introduce defined judgement forms \(\Gamma \vdash F : K\) and \(\Gamma \vdash F = G : K\) for the inhabitation of a product kind \(K\) by an abstraction \(F\), and the equality of two abstractions \(F\) and \(G\) of product kind \(K\); here, \(F\), \(G\) and \(K\) must all have the same arity.

\[
\begin{align*}
\Gamma \vdash [\vec{x}]M : (\Delta)T & \equiv \Gamma, \Delta \vdash M : T \\
\Gamma \vdash [\vec{x}]M = [\vec{x}]N : (\Delta)T & \equiv \Gamma, \Delta \vdash M = N : T
\end{align*}
\]
We assume here that we have applied \( \alpha \)-conversion to ensure that the same variable sequence \( \vec{x} \) is used in both \([\vec{x}]P\) and \([\vec{x}]Q\), and is also the domain of the context \( \Delta \).

Finally, we introduce a judgement form \( \Gamma \vdash \vec{F} :: \Delta \) and \( \Gamma \vdash \vec{F} = \vec{G} :: \Delta \), where \( \vec{F} \) and \( \vec{G} \) are abstraction sequences and \( \Delta \) a context, all of the same arity. These judgement forms are intended to denote that \( \vec{F} \) is a sequence of objects whose kinds are those given by the context \( \Delta \); and that \( \vec{F} \) and \( \vec{G} \) are two such sequences of objects, each member of which is equal to the corresponding member of the other. We say that \( \vec{F} \) satisfies \( \Delta \), and that \( \vec{F} \) and \( \vec{G} \) are two equal sequences that satisfy \( \Delta \).

The judgment forms are defined as follows:

\[
\Gamma \vdash \langle \rangle :: \langle \rangle = \{ \Gamma \text{ valid} \}
\]

\[
\Gamma \vdash (\vec{F}, F_0) :: (\Delta, x : K) = (\Gamma \vdash \vec{F} :: \Delta) \cup \{ \Gamma \vdash F_0 : \{ \vec{F} / \Delta \} K \}
\]

\[
\Gamma \vdash \langle \rangle = \langle \rangle :: \langle \rangle = \{ \Gamma \text{ valid} \}
\]

\[
\Gamma \vdash (\vec{F}, F_0) = (\vec{G}, G_0) :: (\Delta, x : K) = (\Gamma \vdash \vec{F} = \vec{G} :: \Delta) \cup \{ \Gamma \vdash F_0 = G_0 : \{ \vec{F} / \Delta \} K \}
\]

### 2.6 Rules of Deduction

We are finally able to give the rules of deduction of TF. They are listed in Figure 1. They consist of the rules (emp) and (ctxt) determining when a context is valid; (var) and (var_eq), the typing and congruence rules for the application of a variable; (ref), (sym) and (trans), which ensure that the judgemental equality is an equivalence relation; and (conv) and (conv_eq), which ensure that equal kinds have the same objects.

We note in passing how few rules there are compared to logical frameworks of equal expressiveness such as LF [6] and ELF [5]. In particular, the two rules (var) and (var_eq) do all the work normally done by the rules governing typing and congruence of applications and abstractions, and beta- and eta-contractions. We have shifted this burden from the rules of deduction to the syntax. We no longer have to beta- and eta-convert by hand in the derivations; it is now the operation of instantiation that does all the work.

#### 2.6.1 Specifying Object Theories

An object theory is represented in TF by extending the logical framework with several new rules of deduction, representing the formation of the terms and types of the object theory and the computation rules of the object theory.

More formally, an object theory specification in TF is a list of declarations, of two possible forms:

- **constant declarations** of the form

  \[ c : K \]
(emp_ctx) (\{} \text{valid} \quad (ctx) \frac{\Gamma \vdash K \text{ kind}}{\Gamma, x : K \text{ valid}} (x \notin \text{dom} \Gamma)

(var) \quad \frac{\Gamma \vdash \vec{F} :: \Delta}{\Gamma \vdash x \vec{F} : \{\vec{F}/\Delta\}T} (x : (\Delta)T \in \Gamma)

(var_eq) \quad \frac{\Gamma \vdash \vec{F} = \vec{G} :: \Delta}{\Gamma \vdash x \vec{F} = x \vec{G} : \{\vec{F}/\Delta\}T} (x : (\Delta)T \in \Gamma)

(ref) \quad \frac{\Gamma \vdash M : T}{\Gamma \vdash M = M : T} \quad \text{(sym)} \quad \frac{\Gamma \vdash M = N : T}{\Gamma \vdash N = M : T}

(trans) \quad \frac{\Gamma \vdash M = N : T \quad \Gamma \vdash N = P : T}{\Gamma \vdash M = P : T}

(conv) \quad \frac{\Gamma \vdash M : \text{El}(A) \quad \Gamma \vdash A = B : \text{Type}}{\Gamma \vdash M : \text{El}(B)}

(conv_eq) \quad \frac{\Gamma \vdash M = N : \text{El}(A) \quad \Gamma \vdash A = B : \text{Type}}{\Gamma \vdash M = N : \text{El}(B)}

Figure 1: Rules of Deduction of TF

where \(c\) is a constant and \(K\) a kind of the same arity; and

- \textit{equation declarations} of the form

\[(\Delta)(M = N : T)\]

where \(\Delta\) is a context, \(M\) and \(N\) objects and \(T\) a base kind.

The intention is that the constant declarations represent the term- and type-constructors of the object theory, and the equation declarations represent the computation rules of the object theory.

Making the constant declaration \(c : (\Delta)T\) has the effect of adding the following two rules of deduction to the framework (c.f. the rules (var) and (var_eq)):

\[
\begin{align*}
\text{(const)} & \quad \frac{\Gamma \vdash \vec{F} :: \Delta}{\Gamma \vdash c\vec{F} : \{\vec{F}/\Delta\}T} \\
\text{(const_eq)} & \quad \frac{\Gamma \vdash \vec{F} = \vec{G} :: \Delta}{\Gamma \vdash c\vec{F} = c\vec{G} : \{\vec{F}/\Delta\}T}
\end{align*}
\]

(3)

Making the equation declaration \((\Delta)(M = N : T)\) has the effect of adding the following rule to the framework:

\[
\begin{align*}
\text{(eq)} & \quad \frac{\Gamma \vdash \vec{F} :: \Delta}{\Gamma \vdash \{\vec{F}/\Delta\}M = \{\vec{F}/\Delta\}N : \{\vec{F}/\Delta\}T}
\end{align*}
\]

(4)
We shall say that an object theory specification is *coherent* iff the following conditions hold:

- For every constant declaration $c : K$ in the specification, the judgement $\vdash K$ kind is derivable.
- For every equation declaration $(\Delta)(M = N : T)$ in the specification, the judgements $\Delta \vdash M : T$ and $\Delta \vdash N : T$ are derivable.

‘Derivable’ here in both instances means derivable in the logical framework *extended by the specification*; that is, the new rules of deduction (3) and (4) may be used in the derivation.

### 3 Metatheory

We can now begin to investigate the metatheoretical properties of this system. Many of these properties are more difficult to prove than the corresponding properties of a traditional logical framework; in particular, it is often the case that several properties need to be established simultaneously by a single induction. This should be seen as the ‘one-time’ cost of using a lambda-free framework.

#### 3.1 Grammar

We begin by demonstrating the following properties of the operations of instantiation and employment:

The following is the analogue of the result that, if $x$ is not free in $N$, then $\{M/x\}N \equiv N$.

**Lemma 3.1** If $x$ does not occur free in $M$, then

$$\{F/x\}M \equiv M .$$

**Proof** This is easily proven by induction on the object $M$. \hfill \Box

**Lemma 3.2** Let $\alpha$, $\beta$ and $\gamma$ be arities.

1. Let $F$ be an $\alpha$-ary abstraction, $x$ an $\alpha$-ary variable, $G$ a $\beta$-ary abstraction, $y$ a $\beta$-ary variable, and $M$ an object.

   If $x$ and $y$ are distinct variables, and $y$ does not occur free in $M$, then
   $$\{F/x\}\{G/y\}M \equiv \{\{F/x\}G/y\}\{F/x\}M .$$

   (This is the analogue of the famous Substitution Lemma.)

2. Let $F$ be an $\alpha$-ary abstraction, $x$ an $\alpha$-ary variable, $G$ a $(\beta)\gamma$-ary abstraction, and $H$ a $\beta$-ary abstraction. Then

   $$\{F/x\}(G \cdot H) \equiv (\{F/x\}G) \cdot \{F/x\}H .$$
Proof Both parts are proved simultaneously by induction on the sum of the orders of $\alpha$ and $\beta$.

1. This part must be proven by a secondary induction on the object $M$.
   The case where $M \equiv z\bar{H}$ for $z \not\equiv x, y$ is straightforward. We give the calculations for the other two cases:

   \[
   \{F/x\}{G/y}\bar{y}\bar{H} \equiv \{F/x\}(G \bullet \{G/y\}\bar{H})
   \equiv (\{F/x\}G) \bullet \{F/x\}{G/y}\bar{H} \quad \text{(induction hypothesis on arities)}
   \equiv (\{F/x\}G) \bullet \{(F/x)G\}\{F/x\}\bar{H} \quad \text{(induction hypothesis on $P$)}
   \equiv \{(F/x)G/y\}(y(F/x)\bar{H})
   \equiv \{(F/x)G/y\}\{(F/x)y\bar{H}
   \]

   \[
   \{F/x\}{G/y}x\bar{H} \equiv \{F/x\}(x(G/y)\bar{H})
   \equiv F \bullet \{F/x\}{G/y}\bar{H}
   \equiv F \bullet \{(F/x)G/y\}\{F/x\}\bar{H}
   \equiv \{(F/x)G/y\}(F \bullet \{F/x\}\bar{H})
   \quad \text{(induction hypothesis on arities, since $y$ not free in $M$)}
   \equiv \{(F/x)G/y\}\{(F/x)(x\bar{H})
   \]

2. Let $G : [y]G_0$, where we may assume by $\alpha$-conversion that $y \not\equiv x$ and $y$ is not free in $F$. Then

   \[
   \{F/x\}(G \bullet \bar{H}) \equiv \{F/x\}(H/y)G_0
   \equiv \{(F/x)H/y\}\{F/x\}G_0 \quad \text{(from above)}
   \equiv ([y]F_0/x)G_0 \bullet \{F/x\}H
   \equiv (\{F/x\}G) \bullet \{F/x\}H \square
   \]

Lemma 3.3 Let $\alpha$ be an arity.

1. For any $\alpha$-ary variable $x$ and $\alpha$-ary abstraction $F$,
   \[
   \{F/x\}x^\alpha \equiv F .
   \]
   (This is the analogue of the fact that $[M/x]x \equiv M$.)

2. For any $\alpha$-ary variable $x$ and $\alpha$-ary abstraction sequence $\vec{F}$,
   \[
   x^\alpha \bullet \vec{F} \equiv x\vec{F} .
   \]

3. For any $\alpha$-ary variable $x$ and object $M$,
   \[
   \{x^\alpha/x\}M \equiv M .
   \]
   (This is the analogue of the result that $[x/x]M \equiv M$.)
Proof  The three parts are proven simultaneously by induction on \( \alpha \).

1. Let \( F \equiv [\vec{y}]M \). Then

\[
\{F/x\}x^n \equiv \{F/x\}[\vec{y}]x\vec{y}^n
\]
\[
\equiv [\vec{y}](F \bullet \{F/x\}\vec{y}^n)
\]
\[
\equiv [\vec{y}](F \bullet \vec{y}^n) \quad \text{(Lemma 3.1)}
\]
\[
\equiv [\vec{y}](\vec{y}^n/\vec{y})M
\]
\[
\equiv [\vec{y}]M \quad \text{(induction hypothesis)}
\]
\[
\equiv F
\]

2.

\[
x^n \bullet \vec{F} \equiv (([\vec{y}]x\vec{y}^n) \bullet \vec{F}
\]
\[
\equiv \{\vec{F}/\vec{y}\}(x\vec{y}^n)
\]
\[
\equiv x\{\vec{F}/\vec{y}\}\vec{y}^n
\]
\[
\equiv x\vec{F} \quad \text{(by part 1 above)}
\]

3. This part must be proven by a secondary induction on the object \( M \). We deal here with the case that \( M \) has the form \( x\vec{F} \):

\[
\{x^n/x\}x\vec{F} \equiv x^n \bullet \{x^n/x\}\vec{F}
\]
\[
\equiv x^n \bullet \vec{F} \quad \text{(induction hypothesis on \( M \))}
\]
\[
\equiv x\vec{F} \quad \text{(by part 2 above)}
\]

\( \square \)

3.2 Metatheoretic Properties

The following results are true in TF.

**Theorem 3.4 (Context Validity)** Every derivation of a judgement of the form \( \Gamma, \Delta \vdash J \) has a subderivation of \( \Gamma \) valid.

**Proof** A simple induction on derivations. \( \square \)

**Corollary 3.4.1** Every derivation of \( \Gamma, x : K, \Delta \vdash J \) has a subderivation of \( \Gamma \vdash K \) kind.

**Lemma 3.5** If \( \Gamma \vdash J \) is derivable, then every free variable in the judgement body \( J \) is in the domain of \( \Gamma \).

**Proof** A simple induction on derivations. \( \square \)

**Corollary 3.5.1** If \( \Gamma, x : K, \Delta \) valid, then every free variable in \( K \) is in the domain of \( \Gamma \).

**Theorem 3.6 (Weakening)** If \( \Gamma \vdash J, \Gamma \subseteq \Delta \) and \( \Delta \) valid, then \( \Delta \vdash J \).
Proof A simple induction on the derivation of $\Gamma \vdash J$. □

Apart from these four, the metatheoretic properties of TF are very difficult to establish. We relegate the difficult proof of the following properties to the Appendix.

**Theorem 3.7** The following properties hold in TF.

1. **Cut**
   If $\Gamma, x : K, \Delta \vdash J$ and $\Gamma \vDash F : K$ then $\Gamma, \{F/x\} \Delta \vdash \{F/x\} J$.

2. **Functionality**
   If $\Gamma, x : K, \Delta \vdash M : T$ and $\Gamma \vDash F = G : K$ then $\Gamma, \{F/x\} \Delta \vdash \{F/x\} M = \{G/x\} M : \{F/x\} T$.

3. **Context Conversion**
   If $\Gamma, x : K, \Delta \vdash J$ and $\Gamma \vDash K = K'$ then $\Gamma, x : K', \Delta \vdash J$.

4. **Equation Validity**
   If $\Gamma \vdash M = N : T$ then $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$.

**Proof** See Appendix.

Once we have got past this hurdle, other properties of TF follow rapidly.

**Theorem 3.8 (Kind Validity)** The following rules are admissible in TF:

$$
\frac{\Gamma \vDash F : K \quad \Gamma \vDash G : K}{\Gamma \vDash K \text{ kind} \quad \Gamma \vDash K' \text{ kind}}
$$

**Proof** We prove both rules admissible simultaneously by induction on the derivation of the premise. We deal here with the case of the rule

$$(\text{var_eq}) \quad \Gamma \vdash \vec{F} = \vec{G} :: \Delta \quad \Gamma \vdash x\vec{F} = x\vec{G} : \{\vec{F}/\Delta\} T$$

By Equation Validity, $\Gamma \vdash \vec{F} :: \Delta$. By Context Validity,

$$\Gamma, \Delta \vdash T \text{ kind} .$$

The desired judgement $\Gamma \vdash \{\vec{F}/\Delta\} T \text{ kind}$ follows by Cut. □

**Theorem 3.9 (Generation)**

1. If $\Gamma \vdash x\vec{F} : T$, then there is a declaration $x : (\Delta) S$ in $\Gamma$, where

$$\Gamma \vdash \vec{F} :: \Delta, \quad \Gamma \vdash \{\vec{F}/\Delta\} S = T .$$

2. If $\Gamma \vdash c\vec{F} : T$, then a constant declaration $c : (\Delta) S$ has been made, where

$$\Gamma \vdash \vec{F} :: \Delta, \quad \Gamma \vdash \{\vec{F}/\Delta\} S = T .$$

**Proof** Each part is proven by a simple induction on derivations. □

**Corollary 3.9.1** If $\Gamma \vdash M : T$ and $\Gamma \vdash M : T'$, then $\Gamma \vdash T = T'$.

**Proof** Let $M \equiv z\vec{F}$. By Generation, there is a declaration $z : (\Delta) S$ (in $\Gamma$ if $z$ is a variable, or in the object theory specification if $z$ is a constant), where

$$\Gamma \vdash T = T' = \{\vec{F}/\Delta\} S .$$

□
4 The Church-Typed TF

The version of TF we have described is Curry-typed; that is, of the two original versions of the simply-typed lambda calculus introduced by Curry [4] and Church [3], it is most similar to that of Curry, as the bound variables in abstractions are not annotated with their kinds. We can also construct a Church-typed version of TF, in which objects have the form

\[ z[x_{11} : K_{11}, \ldots, x_{1r_1} : K_{1r_1}]M_1, \ldots, [x_{n1} : K_{n1}, \ldots, x_{nr_n} : K_{nr_n}]M_n \].

We shall call the Church-typed version of TF by the name TF\(_k\). In this section, we shall give the definition of TF\(_k\), prove its metatheoretic properties, and define mutually inverse translations between TF and TF\(_k\) that show that the two systems are in some sense equivalent.

It is very convenient to have available two versions of a lambda-free logical framework, and be able to switch between them at will. For example, when embedding a lambda-free frame in a traditional frame, it is easier to define translations into the Curry-typed version, and from the Church-typed version. We shall be in just this situation when we come to embed TF in LF.

Unfortunately, we must prove most of the metatheoretic properties of TF\(_k\) from scratch, repeating a lot of the work we did for TF; once again, the properties must all be established simultaneously by a Herculean induction. We shall give the proof here more briefly than we did in Section 2, as most of the details are similar. There does not seem to be a way to avoid this work; we cannot infer properties of TF\(_k\) from those of TF until we have established the translations between them, and proving these translations sound requires several properties of both systems.

4.1 Grammar

In TF\(_k\), the sets of objects, abstractions, abstraction sequences, contexts and kinds are all defined simultaneously as follows.

**Objects** An object has the form \(z\vec{F}\), where \(z\) is an \(\alpha\)-ary variable or constant and \(\vec{F}\) an \(\alpha\)-ary abstraction sequence, for some arity \(\alpha\).

**Abstractions** An \(\alpha\)-ary abstraction has the form \([\Delta]M\), where \(\Delta\) is an \(\alpha\)-ary context and \(M\) an object.

**Abstraction Sequences** An \((\alpha_1, \ldots, \alpha_n)\)-ary abstraction sequence has the form \(\langle F_1, \ldots, F_n \rangle\), where each \(F_i\) is an \(\alpha_i\)-ary abstraction.

**Contexts** An \((\alpha_1, \ldots, \alpha_n)\)-ary context has the form \(x_1 : K_1, \ldots, x_n : K_n\), where each \(x_i\) is an \(\alpha_i\)-ary variable and \(K_i\) an \(\alpha_i\)-ary kind, with the \(x_i\)s all distinct.

\(^1\)The ‘k’ here stands for ‘kind’, as we include the kind labels in abstractions. This system was named TF\(_c\) in [2], the ‘c’ standing for ‘Church’. I have decided to abandon this name, as ‘c’ could just as well stand for ‘Curry’!
**Kinds** An $\alpha$-ary kind has the form $(\Delta)\text{Type}$ or $(\Delta)\text{El}(M)$, where $\Delta$ is an $\alpha$-ary context and $M$ an object.

In an abstraction $[x_1 : K_1, \ldots, x_n : K_n]M$ or a kind $(x_1 : K_1, \ldots, x_n : K_n)\text{Type}$ or $(x_1 : K_1, \ldots, x_n : K_n)M$, each variable $x_i$ is bound wherever it occurs in $K_{i+1}$, $K_{i+2}$, $\ldots$, $K_n$, and $M$. We identify abstractions and kinds up to $\alpha$-conversion.

The $\eta$-long form of a symbol in $\text{TF}_k$ must be defined with reference to some kind. For $z$ an $\alpha$-ary variable or constant and $K$ an $\alpha$-ary kind, we define the $\alpha$-ary abstraction $z^K$, the $\eta$-long form of $z$ considered as being of kind $K$, by recursion on $\alpha$ as follows.

$$z^{(x_1 : K_1, \ldots, x_n : K_n)T} \equiv [x_1 : K_1, \ldots, x_n : K_n]z[x_1^{K_1}, \ldots, x_n^{K_n}] \cdot$$

The definitions of instantiation and employment in $\text{TF}_k$ are very similar to those in TF.

$${\{F/x\}}z[G_1, \ldots, G_n] \equiv z[\{F/x\}G_1, \ldots, \{F/x\}G_n] \quad (z \not\equiv x)$$

If $F \equiv [t_1 : K_1, \ldots, t_n : K_n]M$, 

$$\{F/x\}x[G_1, \ldots, G_n] \equiv \{\{F/x\}G_1/t_1\} \cdots \{\{F/x\}G_n/t_n\}M$$

$$(\{x : K\}F) \cdot G \equiv \{G/x\}F$$

As in TF, there are three primitive judgement forms in $\text{TF}_k$:

$$\Gamma \vdash M : T$$

where $\Gamma$ is a context, $M$ and $N$ objects and $T$ a base kind.

We define the judgement forms $\Gamma \vdash K$ kind, $\Gamma \vdash K = K'$ and $\Gamma \vdash \Delta = \Delta'$ just as we did for TF. We must take some care over the definitions of the other four defined judgement forms, however.

The judgement form $\Gamma \vdash F : K$, where $F$ is an $\alpha$-ary abstraction and $K$ an $\alpha$-ary kind, is defined as follows.

$$(\Gamma \vdash [\Delta]M : (\Delta')T) = (\Gamma \vdash \Delta' = \Delta)$$

$$\cup \{\Gamma, \Delta' \vdash M : T\}$$

The judgement form $\Gamma \vdash F = G : K$, where $F$ and $G$ are $\alpha$-ary abstractions and $K$ an $\alpha$-ary kind, is defined as follows.

$$(\Gamma \vdash [\Delta_1]M = [\Delta_2]N : (\Delta_3)T) = (\Gamma \vdash \Delta_3 = \Delta_1)$$

$$\cup \{\Gamma, \Delta_3 \vdash M = N : T\}$$

$$(\Gamma \vdash \Delta_3 = \Delta_2)$$

$$\cup \{\Gamma, \Delta_3 \vdash M = N : T\}$$
The judgement form $\Gamma \vdash \vec{F} :: \Delta$, where $\vec{F}$ is an $\alpha$-ary abstraction sequence and $\Delta$ an $\alpha$-ary context, is defined by recursion on $\alpha$ as follows.

$$(\Gamma \vdash \langle \rangle :: \langle \rangle) = \{\text{valid}\}$$

$$(\Gamma \vdash \vec{F}, F_0 :: \Delta, x : K) = (\Gamma \vdash \vec{F} :: \Delta)$$

$$\cup (\Gamma \vdash F_0 : \{\vec{F}/\Delta\} K)$$

The judgement form $\Gamma \vdash \vec{F} = \vec{G} :: \Delta$, where $\vec{F}$ and $\vec{G}$ are $\alpha$-ary abstraction sequences and $\Delta$ an $\alpha$-ary context, is defined by recursion on $\alpha$ as follows.

$$(\Gamma \vdash \langle \rangle = \langle \rangle :: \langle \rangle) = \{\text{valid}\}$$

$$(\Gamma \vdash \vec{F}, F_0 = \vec{G}, G_0 :: \Delta, x : K) = (\Gamma \vdash \vec{F} = \vec{G} :: \Delta)$$

$$\cup (\Gamma \vdash F_0 = G_0 : \{\vec{F}/\Delta\} K)$$

### 4.1.1 Rules of Deduction

The rules of deduction of $\text{TF}_k$ look exactly the same as those of $\text{TF}$, as given in Fig. 1, so we do not reprint them here. We note, however, that the rules (ctxt), (var) and (var_eq) in $\text{TF}_k$ of course use the new definitions of the defined judgement forms $\Gamma \vdash K \text{ kind}$, $\Gamma \vdash \vec{F} :: \Delta$ and $\Gamma \vdash \vec{F} = \vec{G} :: \Delta$.

Object theories are declared in $\text{TF}_k$ in the same way as in $\text{TF}$. We make a number of constant declarations $c : (\Delta)T$, which has the effect of introducing the rules (const) and (const_eq), and equation declarations $(\Delta)(M = N : T)$, which has the effect of introducing the rule (eq), as given in Section 2.6.1.

### 4.2 Metatheory

All the properties of $\text{TF}$ we proved in Section 3 hold in $\text{TF}_k$ too. The proofs follow the same pattern; we have indicated in the Appendix the places where the details differ.

### 4.3 Translations between $\text{TF}$ and $\text{TF}_k$

The systems $\text{TF}$ and $\text{TF}_k$ are equivalent, in the following sense. Given any derivable judgement in $\text{TF}_k$, erasing the kind labels on variables gives a derivable judgement in $\text{TF}$. Conversely, given any derivable judgement in $\text{TF}$, there is a way of filling in the kind labels on the variables to yield a derivable judgement in $\text{TF}$; further, the choice of kind labels is unique up to equality in $\text{TF}_k$.

This fact is very convenient when working with lambda-free logical frameworks, as it allows us to switch between $\text{TF}$ and $\text{TF}_k$ more or less at will, effectively treating them as if they were the same system.

In this section, we shall formally establish the equivalence of $\text{TF}$ and $\text{TF}_k$ by defining translations between the two.

The translation from $\text{TF}_k$ to $\text{TF}$ consists simply of erasing the kind labels:
Definition 4.1 For every object (abstraction, abstraction sequence, kind, context, judgement, type theory specification) \( X \) in \( \text{TF}_k \), let \(|X|\) denote the object (abstraction, abstraction sequence, kind, context, judgement, type theory specification) obtained by erasing the kind labels on the bound variables in abstractions.

Given a type theory specification \( T \) in \( \text{TF}_k \), let \(|T|\) denote the type theory specification in \( \text{TF} \) formed by erasing the kind labels on the bound variables in abstractions within the declarations of \( T \).

It is straightforward to show that this translation is sound:

Theorem 4.2 Let \( T \) be a type theory specification in \( \text{TF}_k \), and let \( J \) be a judgement that is derivable under \( T \). Then \(|J|\) is a derivable judgement in \( \text{TF} \) under the type theory specification \(|T|\).

Proof The proof consists of observing that the image of a primitive rule of deduction in \( \text{TF}_k \) under \(||\) is a primitive rule of deduction in \( \text{TF} \), and the image of any of the rules introduced by \( T \) under \(||\) is a rule introduced by \(|T|\).

Defining the translation in the other direction is harder. We shall define the translation \( 'L' \) from \( \text{TF} \) to \( \text{TF}_k \), which fills in the kind labels on the bound variables. Whenever we encounter an object of the form \( x[\cdots,y_1,\ldots,y_n]M,\cdots] \), we discover the kinds of \( y_1,\ldots,y_n \) by looking up the kind of \( x \) in the current context. Similarly, we handle objects of the form \( c[\cdots] \) by looking up the kind of \( c \) in the object theory specification.

Let us say that an object, abstraction or abstraction sequence \( X \) in \( \text{TF} \) is defined relative to the type theory specification \( T \) and context \( \Gamma \) if and only if every constant that occurs in \( X \) is declared in \( T \), and every free variable in \( X \) is declared in \( \Gamma \). Let us also say that a context \( \Delta \equiv x_1:K_1,\ldots,x_n:K_n \) is defined relative to \( \Gamma \) and \( T \) if and only if, for each \( i \), \( K_i \) is defined relative to the context \( \Gamma, x_1:K_1,\ldots,x_{i-1}:K_{i-1} \) and \( T \); and that the specification \( T \) is consistent if and only if:

- for each constant declaration \( c:K \), \( K \) is defined relative to the empty context and \( T \);
- for each equation declaration \( (\Delta)(M = N:T) \), \( \Delta \) is defined relative to \( T \), and \( M, N \) and \( T \) are defined relative to \( \Delta \) and \( T \).

Now, given a consistent specification \( T \) in \( \text{TF} \), we shall define the following.

- For every context \( \Gamma \) defined relative to \( T \), and every object \( M \) defined relative to \( T \) and \( \Gamma \), an object \( \mathcal{L}_{\Gamma}(M) \) in \( \text{TF}_k \).
- For every abstraction \( F \) and kind \( K \) of the same arity defined relative to \( \Gamma \) and \( T \), an abstraction \( \mathcal{L}_{\Gamma}^K(F) \) in \( \text{TF}_k \). We think of \( K \) as the intended kind of \( F \).
- For every abstraction sequence \( F \) and context \( \Delta \) of the same arity defined relative to \( \Gamma \) and \( T \), an abstraction sequence \( \mathcal{L}_{\Gamma}^\Delta(F) \) in \( \text{TF}_k \). We think of \( \Delta \) as giving the intended kinds of the abstractions \( \bar{F} \).
• For every kind $K$ defined relative to $\Gamma$ and $T$, a kind $L_\Gamma(K)$ in $\text{TF}_k$.
• For every context $\Gamma$ defined relative to $T$, a context $L(\Gamma)$ in $\text{TF}_k$.

The definition is as follows.

If $c : (\Delta)T$ is a declaration in $T$, then

$$L_\Gamma(c \vec{F}) \equiv c[L_\Gamma^\Delta(\vec{F})].$$

If $x : (\Delta)T$ is a declaration in $\Gamma$, then

$$L_\Gamma(x \vec{F}) \equiv x[L_\Gamma^\Delta(\vec{F})].$$

$$L_\Gamma(\langle \rangle) \equiv \langle \rangle$$

$$L_\Gamma(\Gamma, x : K) \equiv L(\Gamma), L_\Gamma(K)$$

We extend the definition to judgements as follows.

$$L(\Gamma \text{ valid}) \equiv L(\Gamma) \text{ valid}$$

$$L(\Gamma \vdash M : T) \equiv L(\Gamma) \vdash L_\Gamma(M) : L_\Gamma(T)$$

$$L(\Gamma \vdash M = N : T) \equiv L(\Gamma) \vdash L_\Gamma(M) = L_\Gamma(N) : L_\Gamma(T)$$

Finally, we extend the definition to type theory specifications. Given a specification $S$ in $\text{TF}$, let $L(S)$ be the following type theory specification in $\text{TF}_k$.

• For every constant declaration $c : K$ in $S$, declare $c : L_\Gamma(K)$.
• For every equation declaration $(\Delta)(M = N : T)$ in $S$, declare $(L(\Delta))(L_\Delta(M) = L_\Delta(N) : L_\Delta(T))$.

We can now show that this translation is sound, after proving a number of lemmas.

**Lemma 4.3** If $M$ is defined relative to both $\Gamma$ and $\Delta$, and $\Gamma$ and $\Delta$ agree on every free variable in $M$, then $L_\Gamma(M) \equiv L_\Delta(M)$. In particular, if $M$ is defined relative to $\Gamma$ and $\Gamma \subseteq \Delta$, then $L_\Gamma(M) \equiv L_\Delta(M)$. 

18
Lemma 4.4 For each of the following equations, if the left-hand side is defined then so is the right-hand side, in which case the two are equal.

\[
\begin{align*}
\{L^K_\Gamma(F)/x\}L_{\Gamma,x:K,\Delta}(M) &= L_{\Gamma,\{F/x\}),(\{F/x\}M) \\
\{L^K_\Gamma(F)/x\}L^K'_{\Gamma,x:K,\Delta}(G) &= L^K_{\Gamma,\{F/x\}),(\{F/x\}G) \\
\{L^K_\Gamma(F)/x\}L^{\Theta}_\Gamma,x:K,\Delta(\vec{G}) &= L^{\Theta}_{\Gamma,\{F/x\}),(\{F/x\}\vec{G}) \\
\{L^K_\Gamma(F)/x\}L_{\Gamma,x:K,\Delta}(K') &= L_{\Gamma,\{F/x\}),(\{F/x\}K') \\
\{L^K_\Gamma(F)/x\}L_{\Gamma,x:K,\Delta}(\Theta) &= L_{\Gamma,\{F/x\}),(\{F/x\}\Theta)
\end{align*}
\]

Proof The five equations are proved simultaneously by a double induction on the arity of $K$, then the size of $M$, $G$, $\vec{G}$, $K'$ and $\Theta$. We give the calculation for one case, the case where $M$ has the form $x\vec{G}$. Let $K \equiv (\Theta)T$ and $F \equiv [\text{dom } \Theta]N$.

\[
\begin{align*}
\{L^K_\Gamma(F)/x\}L_{\Gamma,x:K,\Delta}(x\vec{G}) &= \{L^K_\Gamma(F)/x\}x[L^{\Theta}_\Gamma,x:K,\Delta(\vec{G})] \\
&\equiv L^K_\Gamma(F) \bullet \{L^K_\Gamma(F)/x\}L^{\Theta}_\Gamma,x:K,\Delta(\vec{G}) \\
&\equiv [L_{\Gamma}(\Theta)]L_{\Gamma,\Theta}(N) \bullet L^{\Theta}_\Gamma,(\{F/x\}\vec{G}) \quad \text{(i.h.)} \\
&\equiv \{L^{\Theta}_\Gamma,(\{F/x\}\vec{G})/\Theta\}L_{\Gamma,\Theta}(N) \\
&\equiv \{L^{\Theta}_\Gamma,(\{F/x\}\vec{G})/\Theta\}L_{\Gamma,(\{F/x\}\vec{G})/\Theta}(N) \quad \text{(Lemma 4.3)} \\
&\equiv L_{\Gamma,(\{F/x\}\vec{G})/\Theta}(N) \quad \text{(i.h.)} \\
&\equiv L_{\Gamma,(\{F/x\}x\vec{G})}
\end{align*}
\]

Lemma 4.5

1. If $L_0(\Gamma) \vdash L^K_\Gamma(F) : L_{\Gamma}(K)$, $\vdash L_0(\Gamma) = L_0(\Gamma')$ and $L_0(\Gamma) \vdash L_{\Gamma}(K) = L_{\Gamma'}(K')$, then $L_0(\Gamma) \vdash L^K_\Gamma(F) = L^K_{\Gamma'}(F) : L_{\Gamma}(K)$.

2. If $L_0(\Gamma) \vdash L_{\Gamma}(M) : L_{\Gamma}(T)$ and $\vdash L_0(\Gamma) = L_0(\Gamma')$, then $L_0(\Gamma) \vdash L_{\Gamma}(M) = L_{\Gamma'}(M) : L_{\Gamma}(T)$.

3. If $L_0(\Gamma) \vdash L^{\Theta}_\Gamma(\vec{F}) : L_{\Gamma}(\Theta)$ and $\vdash L_0(\Gamma, \Theta) = L_0(\Gamma', \Theta')$, then $L_0(\Gamma) \vdash L^{\Theta}_\Gamma(\vec{F}) = L^{\Theta'}_{\Gamma'}(\vec{F}) : L_{\Gamma}(\Theta)$.

Proof The three parts are proved simultaneously by induction on $L^K_\Gamma(F)$, $L_{\Gamma}(M)$ and $L^{\Theta}_\Gamma(\vec{F})$. We give here the details for the case in part 2 where $M$ has the form $x\vec{F}$.

Let $x : (\Theta)S \in \Gamma$ and $x : (\Theta')S' \in \Gamma'$. We are given that $L_0(\Gamma) \vdash x\left[L^{\Theta}_\Gamma(\vec{F})\right] : L_{\Gamma}(T)$. Therefore, by Generation,

\[
L_0(\Gamma) \vdash L^{\Theta}_\Gamma(\vec{F}) : L_{\Gamma}(\Theta) \quad L_0(\Gamma) \vdash \{L^{\Theta}_\Gamma(\vec{F})/\Theta\}L_{\Gamma,\Theta}(S) = L_{\Gamma}(T).
\]

19
It also follows from $\vdash L_0(\Gamma) = L_0(\Gamma')$ that

$$L_0(\Gamma) \vdash L_\Gamma(\Theta) = L_{\Gamma'}(\Theta') .$$

Hence, by applying the induction hypothesis to $L_0^\Theta(\vec{F})$, we have

$$L_0(\Gamma) \vdash L_0^\Theta(\vec{F}) = L_{\Gamma'}^\Theta(\vec{F}) :: L_\Gamma(\Theta) .$$

Applying $(\text{var}\.eq)$ then gives us

$$L_0(\Gamma) \vdash x [L_0^\Theta(\vec{F})] = x [L_{\Gamma'}^\Theta(\vec{F})]; \{L_0^\Theta(\vec{F})/\Theta\} L_{\Gamma',\Theta}(S)$$

and the desired result follows using the rule $(\text{conv}\.eq)$. \qed

**Lemma 4.6**

1. If $L(\Gamma \vdash F : K)$ is derivable, then so is $L_0(\Gamma) \vdash L_0^\Theta(\vec{F}) : L_\Gamma(K)$.
2. If $L(\Gamma \vdash \vec{F} :: \Theta)$ are derivable, then so is $L_0(\Gamma) \vdash L_0^\Theta(\vec{F}) :: L_\Gamma(\Theta)$.
3. If $L(\Gamma \vdash F = G : K)$ is derivable, then so is $L_0(\Gamma) \vdash L_0^\Theta(\vec{F}) = L_0^\Theta(\vec{G}) : L_\Gamma(K)$.
4. If $L(\Gamma \vdash \vec{F} = \vec{G} :: \Theta)$ are derivable, then so are $L_0(\Gamma) \vdash L_0^\Theta(\vec{F}) = L_0^\Theta(\vec{G}) :: L_\Gamma(\Delta)$.

**Proof** We give here the details for part 4; the other parts are similar.

Part 4 is proved by induction on the length of $\vec{F}$. If the length is 0, both premise and conclusion are that $L_0(\Gamma)$ is valid.

Suppose now that $\vec{F} = F_0, F_1; \vec{G} = G_0, G_1$; and $\Theta = \Theta_0, x : K_1$. The induction hypothesis gives us that

$$L_0(\Gamma) \vdash L_0^{\Theta_0}(\vec{F}_0) = L_0^{\Theta_0}(\vec{G}_0) ;$$

and by part 3, we have

$$L_0(\Gamma) \vdash L_0^{\Theta_0}(\vec{F}_0) = L_0^{\Theta_0}(\vec{G}_0) ;$$

and

$$L_0(\Gamma) \vdash L_0^{\Theta_0}(\vec{F}_0) = L_0^{\Theta_0}(\vec{G}_0) ;$$

It remains to show that

$$L_0(\Gamma) \vdash L_0^{\Theta_0}(\vec{F}_0) = L_0^{\Theta_0}(\vec{G}_0) ;$$

This follows by the previous lemma from (6) and

$$L_0(\Gamma) \vdash L_0^{\Theta_0}(\vec{F}_0) = L_0^{\Theta_0}(\vec{G}_0) ;$$

which is obtainable using Functionality and (5). \qed

**Theorem 4.7** Let $S$ be a type theory specification in $TF$. Assume we have declared $S$ in $TF$ and $L(S)$ in $TF_k$. Then, for every judgement $J$ derivable in $TF$, the judgement $L(J)$ is derivable in $TF_k$.  

20
Proof The proof is by a straightforward induction on the derivation of $J$. The cases (var), (const), (eq) all make use of Lemma 4.6.2; the cases (var_eq) and (const_eq) make use of Lemma 4.6.4.

Thus, our translations between TF and TF$_k$ are sound. It is also easy to show that the mapping $\mid \mid$ is an exact left inverse to $L$:

Theorem 4.8

$$\mid L_\Gamma (M) \mid \equiv M$$
$$\mid L^K_\Gamma (F) \mid \equiv F$$
$$\mid L^\Delta_\Gamma (\vec{F}) \mid \equiv \vec{F}$$
$$\mid L_\Gamma (K) \mid \equiv K$$
$$\mid L_\Gamma (\Delta) \mid \equiv \Delta$$
$$\mid L(J) \mid \equiv J$$

Proof An easy induction on $M$, $F$, $\vec{F}$, $K$ and $\Delta$. □

The mapping $L$ is not a left inverse to $\mid \mid$ up to syntactic identity. For example,

$$L_{A:Type,B:Type,C:Type}^{(x:El(A))El(C)}(\mid x : El(B) \mid x) \equiv \mid x : El(A) \mid x.$$  

However, on the well-typed objects, abstractions and kinds, we do have that $L$ is a left inverse to $\mid \mid$ up to equality in TF$_k$, in the following sense.

Theorem 4.9 In the framework TF$_k$:

1. If $\Gamma \vdash M : T$ then $\Gamma \vdash M = L_\mid \mid (\mid M \mid) : T$.
2. If $\Gamma \vdash F : K$ then $\Gamma \vdash F = L_\mid \mid^K (\mid F \mid) : K$.
3. If $\Gamma \vdash \vec{F} : \Delta$ then $\Gamma \vdash \vec{F} = L_\mid \mid^\Delta (\mid \vec{F} \mid) : \Delta$.
4. If $\Gamma \vdash K$ kind then $\Gamma \vdash K = L_\mid \mid (\mid K \mid)$.
5. If $\Gamma, \Delta$ valid then $\Gamma \vdash \Delta = L_\mid \mid (\mid \Delta \mid)$.

Proof Let $l(X)$ denote the length of an expression $X$. The five parts are proven simultaneously by induction on $l(\Gamma) + l(M)$, $l(\Gamma) + l(F)$, $l(\Gamma) + l(\vec{F})$, $l(\Gamma) + l(K)$, and $l(\Gamma) + l(\Delta)$. We give here the details of the first two parts.

1. Suppose $M \equiv x\vec{F}$, where $x : (\Theta)S \in \Gamma$. By Generation,

$$\Gamma \vdash \vec{F} : \Theta, \quad \Gamma \vdash \{ \vec{F}/\Theta \} S = T.$$  

Therefore,

$$\Gamma \vdash \vec{F} = L_\mid \mid^{(\Theta)} (\mid \vec{F} \mid) : \Theta \quad \text{(i.h.)}$$

$$\therefore \Gamma \vdash x\vec{F} = x \left[ L_\mid \mid^{(\Theta)} (\mid \vec{F} \mid) \right] : \{ \vec{F}/\Theta \} S \quad \text{(var_eq)}$$

$$\therefore \Gamma \vdash x\vec{F} = x \left[ L_\mid \mid^{(\Theta)} (\mid \vec{F} \mid) \right] : T \quad \text{(conv_eq)}$$
The case $M \equiv c\vec{F}$ is similar.

2. Let $K \equiv (\Theta)T$ and $F \equiv [\Theta']M$. We are given that

$$\Gamma \vdash \Theta = \Theta', \quad \Gamma, \Theta \vdash M : T.$$ 

We must show that $\Gamma \vdash [\Theta']M = [L|_|\Theta] = L|_|(\Theta)T$. The induction hypothesis gives us that $\Gamma, \Theta \vdash M = [L|_|(\Theta)];$ it remains to show

$$\Gamma \vdash \Theta = \Theta' .$$

The induction hypothesis gives us that $\Gamma \vdash [\Theta'] = [L|_|\Theta]$; and $\Gamma \vdash \Theta = [L|_|(\Theta)];$ it is thus sufficient to show

$$\Gamma \vdash L|_|(\Theta) = L|_|(\Theta') .$$

Well,

$$ \vdash \Gamma \vdash L|_|(\Theta) = L|_|(\Theta') \quad \text{(Theorem 4.2)} $$

and the result follows by Context Conversion.

Parts 3–5 are proven similarly. □

5 Embedding TF in LF

Lambda-free frameworks can often be embedded within existing traditional logical frameworks; that is, given a traditional logical framework $F$, we can often construct a lambda-free framework (its core) that is, in some sense, isomorphic to a subsystem of $F$. More precisely, we can construct a lambda-free framework $L$ and define translations $NF$ from $F$ to $L$, and lift from $L$ to $F$. These translations are sound, and $NF$ is a left inverse to lift up to identity ($\alpha$-conversion).

That is, we have the following properties:

1. For every derivable judgement $J$ in $L$, $\text{lift}(J)$ is derivable in $F$.
2. For every derivable judgement $J$ in $F$, $NF(J)$ is derivable in $L$.
3. For every typable expression $X$ in $L$, $NF(\text{lift}(X)) \equiv X$.

In many cases (particularly when $F$ allows $\eta$-conversion) we have in addition that $NF$ is a right inverse to lift up to the equality judgements of $F$:

4. For every typable expression $X$ in $F$, the equality $\text{lift}(NF(X)) = X$ is derivable in $F$. 

22
We can think of $F$ as picking out, from each equivalence class of the expressions of $F$ up to $\beta\eta$-convertibility, a unique representative: the $\beta$-normal, $\eta$-long form.

Establishing the above properties of the translations is not easy; it usually involves proving fairly strong properties of $L$ and $F$. However, once this one-time cost has been paid, we can then use the translations to prove various properties of $F$ more easily. It is often the case that it is easier to establish a given metatheoretic property for $L$ than for $F$. Once it has been proven to hold in $L$, the result can then be "lifted" to $F$; that is, we can derive the corresponding result for $F$ using the properties of the translations.

In this section, we shall show how TF can be embedded in this fashion within the framework LF$^2$ introduced in [6], a Church-typed version of Martin-Löf’s logical framework. It will prove to be very advantageous that we have two different versions of TF; we shall define translations from TF$_k$ to LF, and from LF to TF.

### 5.1 The Framework LF

### 5.2 Translation from TF$_k$ to LF

We shall now define our translations between LF and the two versions of TF. The mapping from TF$_k$ to LF, which we shall call ‘lift’, is defined quite naturally.

We map objects and abstractions to objects, kinds to kinds, contexts to contexts and judgements to judgements as follows.

\[
\begin{align*}
\text{lift}(x|F_1, \ldots, F_n) & \equiv x\text{lift}(F_1) \cdots \text{lift}(F_n) \\
\text{lift}([\Delta]M) & \equiv [\text{lift}(\Delta)]\text{lift}(M) \\
\text{lift}(\text{Type}) & \equiv \text{Type} \\
\text{lift}(\text{El}(M)) & \equiv \text{El}(\text{lift}(M)) \\
\text{lift}((x : K)K') & \equiv (x : \text{lift}(K))\text{lift}(K') \\
\text{lift}(x_1 : K_1, \ldots, x_n : K_n) & \equiv x_1 : \text{lift}(K_1), \ldots, x_n : \text{lift}(K_n) \\
\text{lift}(\Gamma \text{ valid}) & \equiv \text{lift}(\Gamma) \text{ valid} \\
\text{lift}(\Gamma \vdash M : T) & \equiv \text{lift}(\Gamma) \vdash \text{lift}(M) : \text{lift}(T) \\
\text{lift}(\Gamma \vdash M = N : T) & \equiv \text{lift}(\Gamma) \vdash \text{lift}(M) = \text{lift}(N) : \text{lift}(T)
\end{align*}
\]

It is relatively straightforward to establish that this translation is sound.

**Lemma 5.1**

\[
[lift(F)/x]lift(N) \rightarrow_\beta lift\{\{F/x\}N\}
\]

\footnote{The framework here named LF should not be confused with the Edinburgh Logical Framework, ELF [5], which is also sometimes called LF.}
Proof The proof is by a double induction on the arity of $F$ and $x$, then on the object $N$. We give here the details for the case $N \equiv x\vec{G}$. Let $F \equiv [\Delta]P$.

\[
\begin{align*}
\text{lift(}(F/x)\text{)} &= \text{lift(}(F/x)\text{)}
\end{align*}
\]

Lemma 5.2 In LF,

1. If lift($\Gamma \vdash F : K$) are derivable, then so is lift($\Gamma$) $\vdash$ lift($F$) : lift($K$).

2. If lift($\Gamma \vdash F = G : K$) are derivable, then so is lift($\Gamma$) $\vdash$ lift($F$) = lift($G$) : lift($K$).

3. If lift($\Gamma \vdash \vec{F} :: \Delta$) and lift($\Gamma, \Delta$ valid) are derivable, then so are lift($\Gamma$) $\vdash$ lift($\vec{F}$) :: lift($\Delta$).

4. If lift($\Gamma \vdash \vec{F} = \vec{G} :: \Delta$) and lift($\Gamma, \Delta$ valid) are derivable, then so are lift($\Gamma$) $\vdash$ lift($\vec{F}$) = lift($\vec{G}$) :: lift($\Delta$).

Proof

1. Let $K \equiv (\Delta)T$ and $F \equiv [\Delta']M$. We are given lift($\Gamma \vdash [\Delta']M$) and lift($\Gamma$), lift($\Delta$) $\vdash$ lift($M$) : lift($T$). From the first of these, it follows simply that lift($\Gamma$) $\vdash$ lift($\Delta$) = lift($\Delta'$). From this and the second, we can derive that lift($\Gamma$) $\vdash$ lift($\Delta'$) : lift($\Delta$).

2. Similar.

3. The proof is by induction on the length of $\vec{F}$.

If the length is 0, both hypothesis and conclusion are that lift($\Gamma$) is valid.

Suppose $\vec{F}$ is of length $n+1$, and the result holds for abstraction sequences of length $n$. Let $\vec{F} \equiv F_0, F_1$; and $\Delta \equiv \Delta_0, x : K_1$. We are given that lift($\Gamma \vdash F_0 :: \Delta_0$) is derivable, hence so is lift($\Gamma$) $\vdash$ lift($F_0$) :: lift($\Delta_0$) by the induction hypothesis. We also have

\[
\text{lift(}(\Gamma) \vdash \text{lift($F_1$)} : \text{lift($\{ F_0/\Delta_0 \}$)}K_1)
\]

by part 1 and

\[
\text{lift(}(\Gamma), \text{lift($\Delta_0$)} \vdash \text{lift($K_1$)} \text{ kind}
\]
by Kind Validity in LF. This yields
\[ \text{lift}(\Gamma) \vdash [\text{lift}(\vec{F}_0)/\Delta_0]\text{lift}(K_1) \text{ kind} \] (substitution)
\[ \vdash \text{lift}(\Gamma) \vdash [\text{lift}(\vec{F}_0)/\Delta_0]\text{lift}(K_1) = \text{lift}\left\{ \text{lift}(\vec{F}_0)/\Delta_0\right\} \text{lift}(K_1) \] (Subject Reduction, Lemma 5.1)
\[ \therefore \text{lift}(\Gamma) \vdash \text{lift}(F_1) : [\text{lift}(\vec{F}_0)/\Delta_0]\text{lift}(K_1) \] (conv)
as required.

4. Similar. \(\Box\)

**Theorem 5.3** Suppose we have declared a type theory \(T\) in \(TF_k\), and the corresponding theory \(\text{lift}(T)\) in LF. If \(J\) is a derivable judgement in \(TF_k\), then \(\text{lift}(J)\) is derivable in LF.

**Proof** The proof is by induction on the derivation of \(J\). All cases are straightforward using the previous two lemmas. \(\Box\)

### 5.3 Translation from LF to TF

The translation from LF to TF is more difficult to construct. It consists of reducing every entity of LF to its \(\beta\)-normal, \(\eta\)-long form.

We must first assign arities to the entities of LF to guide us during \(\eta\)-expansion. We assign an arity to every kind of LF as follows:

\[
\begin{align*}
\text{Ar}(\text{Type}) & \equiv 0 \\
\text{Ar}(\text{El}(k)) & \equiv 0 \\
\text{Ar}(x : K_1)K_2 & \equiv (\text{Ar}(K_1))\beta \text{Ar}(K_2)
\end{align*}
\]

We now define an arity \(\text{Ar}_\Gamma(k)\) to some LF-contexts \(\Gamma\) and LF-objects \(k\) as follows:

- If \(x : K\) is an entry in \(\Gamma\), then \(\text{Ar}_\Gamma(x) \equiv \text{Ar}(K)\).
- If \(c\) has been declared with arity \(K\), then \(\text{Ar}_\Gamma(c) \equiv \text{Ar}(K)\).
- If \(\text{Ar}_{\Gamma,x,K}(k)\) is defined, then \(\text{Ar}_{\Gamma}([x : K]k) \equiv (\text{Ar}(K))\beta \text{Ar}_{\Gamma,x,K}(k)\).
- If \(\text{Ar}_\Gamma(k)\) and \(\text{Ar}_\Gamma(k')\) is defined, and \(\text{Ar}_\Gamma(k)\) has the form
  \[ \text{Ar}_\Gamma(k) \equiv (\text{Ar}_\Gamma(k'))\beta \]
  then \(\text{Ar}_\Gamma(kk') \equiv \beta\).

We shall say that an object is **well-arithed** if \(\text{Ar}_\Gamma(k)\) is defined. We shall only be able to map well-arithed objects into TF. We can prove immediately that every object typable in LF is well-arithed.

**Proposition 5.4** Let \(T\) be a type theory specification in LF. Suppose that, for every equation declaration \((\Delta)(k = k' : K)\) in \(T\), we have \(\text{Ar}_\Delta(k) \equiv \text{Ar}_\Delta(k') \equiv \text{Ar}(K)\). Then

25
1. if \( \Gamma \vdash k : K \) then \( Ar_{\Gamma}(k) \equiv Ar(K) \);
2. if \( \Gamma \vdash k = k' : K \) then \( Ar_{\Gamma}(k) \equiv Ar_{\Gamma}(k') \equiv Ar(K) \);
3. if \( \Gamma \vdash K = K' \) then \( Ar(K) \equiv Ar(K') \).

**Proof**  The three statements are proven simultaneously by induction on the derivation of the premise. We need to make use of the following two auxiliary facts, which are easy to prove:

1. \( Ar([k/x]K) \equiv Ar(K) \)
2. If \( Ar_{\Gamma}(k) \equiv Ar(K) \) and \( Ar_{\Gamma,x:K}(k') \) is defined, then \( Ar_{\Gamma}([k/x]k') \equiv Ar_{\Gamma,x:K}(k') \).

\[ \square \]

Given an object \( k \) such that \( Ar_{\Gamma}(k) \equiv \alpha \), we define the \( \alpha \)-ary abstraction \( NF_{\Gamma}(k) \) in TF as follows.

\[
NF_{\Gamma}(x) \equiv x^\eta \\
NF_{\Gamma}(c) \equiv c^\eta \\
NF_{\Gamma}([x : K]k) \equiv [x]NF_{\Gamma,x:K}(k) \\
NF_{\Gamma}(k'k') \equiv NF_{\Gamma}(k) \bullet NF_{\Gamma}(k')
\]

where, in the first two clauses, \( x \) has arity \( Ar_{\Gamma}(x) \) and \( c \) has arity \( Ar_{\Gamma}(c) \). In the third clause, \( x \) has arity \( Ar_{\Gamma}(K) \).

We extend the mapping \( NF \) to kinds, contexts and judgements as follows.

\[
NF_{\Gamma}(Type) \equiv Type \\
NF_{\Gamma}(El(k)) \equiv El(NF_{\Gamma}(k)) \\
NF_{\Gamma}((x : K)K') \equiv (x : NF_{\Gamma}(K))NF_{\Gamma,x:K}(K')
\]

\[
NF_{\Gamma}([]) \equiv [] \\
NF_{\Gamma}(\langle \rangle, x : K) \equiv NF_{\Gamma}(\langle \rangle, x : NF_{\Gamma}(K))
\]

The following results ensure that this translation is well-behaved and sound.

**Theorem 5.5** 1. Let \( Ar(K) \equiv \alpha \). If \( NF_{\Gamma}(K) \) is defined, then it is an \( \alpha \)-ary kind.

2. Let \( \Gamma \subseteq \Delta \). If \( NF_{\Gamma}(X) \) is defined, then \( NF_{\Delta}(X) \) is defined, and

\[
NF_{\Delta}(X) \equiv NF_{\Gamma}(X)
\]
3. Suppose \( \text{Ar}_\Gamma(k) \equiv \text{Ar}(K) \). Let \( X \) be an LF-object, kind or context. If \( \text{NF}_\Gamma(k) \) and \( \text{NF}_{\Gamma,x,K,\Delta}(X) \) are defined, then \( \text{NF}_{\Gamma,[k/x],\Delta}([k/x]X) \) is defined, and

\[
\text{NF}_{\Gamma,[k/x],\Delta}([k/x]X) \equiv \{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K,\Delta}(X)
\]

4. If the judgement \( \mathcal{J} \) is derivable in LF, then \( \text{NF}(\mathcal{J}) \) is defined and derivable in TF.

**Proof** The first two parts are easily proven by an induction on \( K \) and \( X \) respectively.

The third part is proven by induction on \( X \). We give the calculation here for the case \( X \equiv x \):

\[
\{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K,\Delta}(x) \equiv \{\text{NF}_\Gamma(k)/x\}x''
\]

\[
\equiv \text{NF}_\Gamma(k) \quad \text{(Lemma \ref{lemma:nf-def})}
\]

\[
\equiv \text{NF}_{\Gamma,[k/x],\Delta}([k/x]x) \quad \text{(part 2)}
\]

The fourth part is proven by induction on the derivation of \( \mathcal{J} \). Most cases are straightforward, making use of the results proven in Section 3. We give the details for the two rules (beta) and (eta).

**(beta)**

\[
\frac{\Gamma, x : K \vdash k' : K' \quad \Gamma \vdash k : K}{\Gamma \vdash ([x : K]k')k = [k/x]k' : [k/x]K'}
\]

By the induction hypothesis,

\[
\text{NF}_\emptyset(\Gamma), x : \text{NF}_\Gamma(K) \vdash \text{NF}_{\Gamma,x,K}(k') : \text{NF}_{\Gamma,x,K}(K') ; \text{NF}_\emptyset(\Gamma) \vdash \text{NF}_\Gamma(k) : \text{NF}_\Gamma(K).
\]

Now,

\[
\text{NF}_\Gamma(([x : K]k')k) \equiv ([x]\text{NF}_{\Gamma,x,K}(k')) \bullet \text{NF}_\Gamma(k)
\]

\[
\equiv \{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K}(k')
\]

\[
\equiv \text{NF}_\Gamma([k/x]k') \quad \text{(part 3)}
\]

The Cut rule and (ref) give us

\[
\text{NF}_\emptyset(\Gamma) \vdash \{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K}(k') = \{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K}(k') : \{\text{NF}_\Gamma(k)/x\}\text{NF}_{\Gamma,x,K}(K')
\]

and, by part 3, this is the same judgement as

\[
\text{NF}_\emptyset(\Gamma) \vdash \text{NF}_\Gamma(([x : K]k')k) = \text{NF}_\Gamma([k/x]k') : \text{NF}_\Gamma([k/x]K').
\]

**(eta)**

\[
\frac{\Gamma \vdash k : (x : K)K'}{\Gamma \vdash k = [x : K]kx : (x : K)K'}
\]

By the induction hypothesis,

\[
\text{NF}_\emptyset(\Gamma) \vdash \text{NF}_\Gamma(k) : (x : \text{NF}_\Gamma(K))\text{NF}_{\Gamma,x,K}(K').
\]
Now,
\[
NF_\Gamma([x : K] k x) \equiv [x] NF_\Gamma, x : K(k x) \\
\equiv [x] NF_\Gamma, x : K(k) \cdot x^n \\
\equiv NF_\Gamma, x : K(k) \quad \text{(Lemma 3.3)} \\
\equiv NF_\Gamma(k) \quad \text{(part 2)}
\]

Thus, (ref) gives us
\[
NF_\emptyset(\Gamma) \vdash NF_\Gamma(k) = NF_\Gamma([x : K] k x) : (x : NF_\Gamma(K)) NF_\Gamma, x : K(K') . \quad \Box
\]

The translations we have established between our three systems are shown in Figure 2. The triangles in this diagram commute in the sense given by the following theorem.

**Theorem 5.6**

1. If \( \Gamma \vdash k : K \) in LF, then
\[
\Gamma \vdash k = lift(L_{NF_\emptyset(\Gamma)}(NF_\Gamma(k))) : K .
\]

Similar results hold for kinds and contexts.

2. If \( \Gamma \vdash M : T \) in TF, then
\[
M \equiv NF_{\text{lift}(L_{\Gamma}(M))}(L_{\Gamma}(M)) .
\]

Similar results hold for kinds and contexts.

3. If \( \Gamma \vdash M : T \) in TF\(_k\), then
\[
\Gamma \vdash M = L_{NF_\emptyset(\text{lift}(\Gamma))}(NF_{\text{lift}(\Gamma)}(\text{lift}(M))) : T .
\]

Similar results hold for kinds and contexts.

**Proof**

1. We prove the statement:

   If \( \Gamma \vdash k : K \) and \( \vdash \Gamma = \Delta \) in LF, then
\[
\Gamma \vdash k = lift(L_{NF_\emptyset(\Gamma)}(NF_\Delta(k))) : K .
\]
We prove the statements simultaneously with similar statements for kinds and contexts by induction on size. We give here the details for the case where \( k \) is an abstraction.

Let \( k \equiv [x : K_0]k' \), and \( K \equiv (x : K_1)K_2 \). By Generation, we have

\[
\Gamma \vdash K_0 = K_1, \quad \Gamma, x : K_1 \vdash k' : K_2 .
\]

Now,

\[
lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(NF\Delta(k))) \equiv \lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x : NF\Gamma(K_1))NF_{\eta,K_1}(K_2)(x[NF\Delta,x : K_0(k')])))
\]

\[
\equiv \lift([x : \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(NF\Gamma(K_1))][\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(\Gamma,x : K_1)(NF\Delta,x : K_0(k')))])
\]

\[
\equiv [x : \lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(NF\Gamma(K_1))))\lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(\Gamma,x : K_1)(NF\Delta,x : K_0(k')))])
\]

Now, the induction hypothesis gives the two judgements

\[
\Gamma \vdash \lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(NF\Gamma(K_1)))) = K_1
\]

\[
\Gamma, x : K_1 \vdash \lift(\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(\Gamma,x : K_1)(NF\Delta,x : K_0(k'))) : K_2
\]

from which the result follows.

2. The proof is by induction on the object \( M \). Let \( M \equiv z[\vec{F}] \), and let \( z \) have kind \((\Delta)T\) relative to \( \Gamma \). Then

\[
NF_{\lift(0)(\Gamma)}(\lift(\mathcal{L}_\Gamma(M))) \equiv NF_{\lift(0)(\Gamma)}(\lift(z[\lift(0)(\vec{F})]))
\]

\[
\equiv \lift(\mathcal{L}_\Gamma(\vec{F}))(\lift(\mathcal{L}_\Gamma(\vec{F})))\]

\[
\equiv \lift(\mathcal{L}_\Gamma(\vec{F}))\]

\[
\equiv \lift(\mathcal{L}_\Gamma(\vec{F}))\]

\[
\equiv \lift(\mathcal{L}_\Gamma(\vec{F}))\]

\[
\equiv \lift(\mathcal{L}_\Gamma(\vec{F}))\]

3. The proof is by induction on the object \( M \). Let \( M \equiv x[\vec{F}] \), and let

\[
\Gamma \equiv \Gamma_1, x : (\Delta)S, \Gamma_2 .
\]

Then

\[
\mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(\lift(M)) \equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(\lift(M))\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]

\[
\equiv \mathcal{L}_{NF,(\Gamma)}^{NF,(\Gamma)}(x[\lift(\vec{F})])\]
Now, by Generation, $\Gamma \vdash F : \Delta$ and $\Gamma \vdash \{\vec{F}/\Delta\}S = T$. Hence, the induction hypothesis gives

$$\Gamma \vdash \vec{F} = \mathcal{L}_{\text{NF}}^\mathcal{L}(\Gamma)(\mathcal{L}(\Gamma)(\mathcal{L}(\Gamma)(\vec{F})) : \Delta$$

from which the result follows.

5.4 Lifting Results

Suppose we wish to establish a property of a framework, or of an object theory in a traditional framework $F$. It is often the case that the property is more easily proven for a lambda-free framework $L$. The result can then be ‘lifted’ to $F$; that is, we can derive the result for $F$ easily from $L$, together with the properties of the translations between $L$ and $F$.

To illustrate this situation, we shall prove the following result about LF, using TF:

**Theorem 5.7 (Injectivity of Type Constructors)** Let $\mathcal{S}$ be an object theory specification in LF that has the property: for every equation declaration $(\Delta)(M = N : T)$ in $\mathcal{S}$, $T$ has the form $\text{El}(A)$ (that is, there are no equation declarations of the form $(\Delta)(M = N : \text{Type})$). Let $c : (\Theta)\text{Type}$ be a constant declaration in $\mathcal{S}$. Then the following rule of deduction is admissible:

$$\Gamma \vdash c\vec{A} = c\vec{B} : \text{Type} \quad \frac{\Gamma \vdash A = B : \Theta}{\Gamma \vdash \vec{A} = \vec{B} : \Theta}$$

The corresponding result for TF is fairly easy to prove:

**Theorem 5.8** Let $\mathcal{S}$ be an object theory specification in TF that has the property: for every equation declaration $(\Delta)(M = N : T)$ in $\mathcal{S}$, $T$ has the form $\text{El}(A)$. Let $c : (\Theta)\text{Type}$ be a constant declaration in $\mathcal{S}$. Then the following rule of deduction is admissible:

$$\Gamma \vdash c\vec{F} = c\vec{G} : \text{Type} \quad \frac{\Gamma \vdash \vec{F} = \vec{G} : \Theta}{\Gamma \vdash \vec{F} = \vec{G} : \Theta}$$

**Proof** We shall prove the following statement.

If $\Gamma \vdash c\vec{F} = X : \text{Type}$ or $\Gamma \vdash X = c\vec{F} : \text{Type}$ is derivable, then $X$ has the form $c\vec{G}$, and $\Gamma \vdash \vec{F} = \vec{G} : \Theta$.

The proof is by induction on the derivation of the premise. Note that the last step in this derivation cannot be the use of an equation from $\mathcal{S}$. All cases are straightforward.

The result can now be ‘lifted’ to LF. We omit the sub- and superscripts on NF and $\mathcal{L}$ in the following proof.
Proof of Theorem 5.7. Let $\mathcal{S}$ satisfy the hypotheses of the theorem. Suppose $\Gamma \vdash c\vec{A} = c\vec{B} : \text{Type}$ is derivable in LF under $\mathcal{S}$. By Theorem 5.5

$$\text{NF}(\Gamma) \vdash c\text{NF}(\vec{A}) = c\text{NF}(\vec{B}) : \text{Type}$$

is derivable in TF under NF($\mathcal{S}$). We note also that NF($\mathcal{S}$) satisfies the hypotheses of Theorem 5.8. Therefore,

$$\text{NF}(\Gamma) \vdash_{TF} \text{NF}(\vec{A}) = \text{NF}(\vec{B}) :: \text{NF}(\Theta) \quad (\text{Theorem } 5.8)$$

We also have, by Theorem 5.6

$$\Gamma \vdash_{LF} \text{lift}(\mathcal{L}(\text{NF}(\Gamma)))$$

$$\Gamma \vdash_{LF} \vec{A} = \text{lift}(\mathcal{L}(\text{NF}(\vec{A}))) :: \Theta$$

$$\Gamma \vdash_{LF} \vec{B} = \text{lift}(\mathcal{L}(\text{NF}(\vec{B}))) :: \Theta$$

$$\Gamma \vdash_{LF} \Theta = \text{lift}(\mathcal{L}(\text{NF}(\Theta)))$$

It follows that

$$\Gamma \vdash_{LF} \vec{A} = \vec{B} :: \Theta$$

as required.

In contrast, the author has been unable to find a direct proof of this result in LF. (Note that, as we do not restrict the equations in $\mathcal{S}$ in any way beyond the hypothesis of the theorem, it may not be the case that the Church-Rosser property holds under $\mathcal{S}$.)

6 Conclusion

References

[1] Peter Aczel. Yet another logical framework. Unpublished.

[2] Robin Adams. A Modular Hierarchy of Logical Frameworks. PhD thesis, University of Manchester, 2004.

[3] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[4] H. B. Curry and R. Feys. Combinatory Logic, volume 1 of Studies in Logic and the Foundations of Mathematics. North Holland, 1948.

[5] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993. A preliminary version appeared in the Proceedings of the Symposium on Logic in Computer Science, pages 194–204, June 1987.
A Metatheory of TF

We present here the proof of the basic metatheoretic properties of TF and TF$_k$. We do so in the following manner. We define a different version of each system, which we name TF' and TF'$_k$. We establish that the desired properties hold of TF' and TF'$_k$; this requires proving several properties simultaneously in one large induction. We then prove that TF is equivalent to TF', and TF$_k$ equivalent to TF'$_k$.

The definition of TF' and TF'$_k$ require only two changes from TF and TF$_k$. Firstly, we replace the rule (var_eq) in Fig. 1 with

\[
\frac{\Gamma \vdash \vec{F} = \vec{G} :: \Delta}{\Gamma \vdash x\vec{F} = x\vec{G} : \{\vec{F}/\Delta\}T} \quad (x : (\Delta)T \in \Gamma)
\]

Secondly, whenever we declare a constant $c : (\Delta)T$, we replace the rule (const_eq) with

\[
\frac{\Gamma \vdash \vec{F} = \vec{G} :: \Delta}{\Gamma \vdash c\vec{F} = c\vec{G} : \{\vec{F}/\Delta\}T}
\]

In this appendix, we shall establish that the following rules of deduction are admissible in TF' and TF'$_k$. The proofs for the two systems are very similar; we shall give the proof for TF', and note the places where changes need to be made for TF'$_k$; we shall mark these places with the symbol §.

\[
\begin{align*}
\text{(cut)} & \quad \frac{\Gamma, x : K, \Delta \vdash J \quad \Gamma \vdash F : K}{\Gamma, \{F/x\} \Delta \vdash \{F/x\} J} \\
\text{(func)} & \quad \frac{\Gamma, x : K, \Delta \vdash M : T \quad \Gamma \vdash F = G : K}{\Gamma, \{F/x\} \Delta \vdash \{F/x\} M = \{G/x\} M : \{F/x\} T} \\
\text{(Rfunc)} & \quad \frac{\Gamma, x : K, \Delta \vdash M : T \quad \Gamma \vdash F = G : K}{\Gamma, \{F/x\} \Delta \vdash \{G/x\} M : \{F/x\} T} \\
\text{(context\_conversion)} & \quad \frac{\Gamma, x : K, \Delta \vdash J \quad \Gamma \vdash K = K'}{\Gamma, x : K', \Delta \vdash J}
\end{align*}
\]
We shall be proving these rules admissible simultaneously by one large induction on the arity of the kind $K$. To this end, it shall be convenient to make the following definitions:

**Definition A.1** Let $\alpha$ be an arity.

1. The rule of deduction (cut$_\alpha$) consists of all the instances of the rule (cut) in which the kind $K$ has arity $\alpha$. We similarly define the rules of deduction (func$_\alpha$), (Rfunc$_\alpha$) and (context$_\alpha$ conversion$_\alpha$); in each case, the rule consists of all those instances of the similarly named rule in which the kind $K$ has arity $\alpha$.

2. $\mathcal{R}(\alpha)$ is the following set of rules of deduction:

$$\mathcal{R}(\alpha) = \{(cut_\alpha), (func_\alpha), (Rfunc_\alpha), (context_\alpha conversion_\alpha)\}.$$

3. A judgement $J$ is $\alpha$-good if, for every rule $R \in \mathcal{R}(\alpha)$, if $J$ is the first premise of an instance of $R$, then that instance is admissible.

Thus, we aim to show that every derivable judgement is $\alpha$-good for every arity $\alpha$.

**Lemma A.2** Let $\alpha$ be an arity. Suppose that (context$_\beta$ conversion$_\beta$) is admissible for every proper subarity $\beta$ of $\alpha$. Then the following are admissible.

$$(Ksym_\alpha) \quad \Gamma \vdash K = K' \quad \frac{}{\Gamma \vdash K' = K}$$

$$(Csym_\alpha) \quad \Gamma \vdash \Delta = \Delta' \quad \frac{}{\Gamma \vdash \Delta' = \Delta}$$

where $K$ and $\Delta$ have arity $\alpha$.

**Proof** The proof is by induction on $\alpha$.

For (Csym$_\alpha$), if $\alpha \equiv 0$, there is nothing to prove. Otherwise, let $\Delta \equiv \Delta_0, x : K$ and $\Delta' \equiv \Delta'_0, x : K'$. We are given

$$\Gamma \vdash \Delta_0 = \Delta'_0; \quad \Gamma, \Delta_0 \vdash K = K'$$

and must show

$$\Gamma \vdash \Delta'_0 = \Delta_0; \quad \Gamma, \Delta'_0 \vdash K' = K.$$

The first of these is given by the induction hypothesis. For the second, applying (context$_\alpha$ conversion) with each of the entries in $\Delta_0$ gives

$$\Gamma, \Delta'_0 \vdash K = K'$$

The result follows by the induction hypothesis.

For (Ksym$_\alpha$), let $K \equiv (\Delta)T$ and $K' \equiv (\Delta')T'$. We are given

$$\Gamma \vdash \Delta = \Delta'; \quad \Gamma, \Delta \vdash T = T'$$

33
and must show
\[ \Gamma \vdash \Delta' = \Delta; \quad \Gamma, \Delta' \vdash T' = T. \]
The first of these is given by (Csym_\alpha), which we have just shown to be admissible. For the second applying (context_conversion) with each of the entries in \( \Delta \) gives \( \Gamma, \Delta' \vdash T' = T \). The result follows by (sym). □

**Lemma A.3** Let \( \alpha \) be an arity. Suppose that (context_conversion_\beta) is admissible for every proper subarity \( \beta \) of \( \alpha \). Then the following are admissible.

\[
\frac{\Gamma \vdash K_1 = K_2}{\Gamma \vdash K_1 = K_3}
\]

\[
\frac{\Gamma \vdash \Delta_1 = \Delta_2}{\Gamma \vdash \Delta_1 = \Delta_3}
\]

where \( K_1 \) and \( \Delta_1 \) have arity \( \alpha \).

**Proof** The proof is by induction on \( \alpha \). We give the details for (Ctrans_\alpha) here; those for (Ktrans_\alpha) are similar.

If \( \alpha \equiv 0 \), there is nothing to prove. If \( \alpha \equiv \alpha' (\beta) \), let \( \Delta_i \equiv \Delta'_i, x : K_i \), for \( i = 1, 2, 3 \). Then we are given

\[ \Gamma \vdash \Delta_i = \Delta'_i \]
\[ \Gamma, \Delta_i \vdash T = T' \]

\[ \Gamma, \Delta_i \vdash K_1 = K_2 \]
\[ \Gamma, \Delta_i \vdash K_2 = K_3 \]

\[ \Gamma \vdash \Delta'_i = \Delta'_3 \]

follows by the induction hypothesis. We also have \( \Gamma, \Delta'_i \vdash K_2 = K_3 \) by (Ksym_\alpha'), and \( \Gamma, \Delta'_i \vdash K_1 = K_3 \) follows by the induction hypothesis. □

**Lemma A.4** Let \( \alpha \) be an arity. Suppose that (context_conversion_\beta) is admissible for every proper subarity \( \beta \) of \( \alpha \). Then the following are admissible.

\[
\frac{\Gamma \vdash F : K \quad \Gamma \vdash K = K'}{\Gamma \vdash F : K'}
\]

\[
\frac{\Gamma \vdash F = G : K \quad \Gamma \vdash K = K'}{\Gamma \vdash F = G : K'}
\]

where \( F, G, K \) and \( K' \) have arity \( \alpha \).

**Proof** We give the details here for (conversion_\alpha); those for (conversion_eq_\alpha) are similar.

Let \( K \equiv (\Delta)T, K' \equiv (\Delta')T', \) and \( F \equiv [\text{dom } \Delta]M \). We are given that

\[ \Gamma, \Delta \vdash M : T, \quad \Gamma \vdash \Delta = \Delta', \quad \Gamma, \Delta \vdash T = T' \]

and must show that \( \Gamma, \Delta' \vdash M : T' \). We have

\[ \Gamma, \Delta' \vdash M : T, \quad \Gamma, \Delta' \vdash T = T' \]

by applying the appropriate (context_conversion_\beta) rules; the result follows by (conv).
§When working in TF, let $K \equiv (\Delta)T$, $K' \equiv (\Delta')T'$, and $F \equiv [\Delta_0]M$. We are given that

$$
\Gamma, \Delta \vdash M : T, \quad \Gamma \Vdash \Delta = \Delta_0, \quad \Gamma \Vdash \Delta = \Delta', \quad \Gamma, \Delta \Vdash T = T'
$$

and must show that $\Gamma, \Delta' \vdash M : T'$ and $\Gamma \Vdash \Delta' = \Delta_0$. The first of these follows as above; the second by (Csym$_\alpha$) and (Ctrans$_\alpha$).

**Lemma A.5** Let $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$. If (cut$_{\alpha_1}$), $\ldots$, (cut$_{\alpha_n}$) are admissible, so is

$$
(Cut_\alpha) \quad \frac{\Gamma, \Delta, \Theta \vdash J \quad \Gamma \Vdash \vec{F} :: \Delta}{\Gamma, \{\vec{F}/\Delta\}\Theta \vdash \{\vec{F}/\Delta\}J}
$$

where $\Delta$ has arity $\alpha$.

**Proof** Simply apply the $n$ rules (cut$_{\alpha_1}$), $\ldots$, (cut$_{\alpha_n}$) in succession to the first premise.

Let (Func$_\alpha$) be the rule

$$
(Func_\alpha) \quad \frac{\Gamma, \Delta, \Theta \vdash M : T \quad \Gamma \Vdash \vec{F} = \vec{G} : \Delta}{\Gamma, \{\vec{F}/\Delta\}\Theta \vdash \{\vec{F}/\Delta\}M = \{\vec{G}/\Delta\}M : \{\vec{F}/\Delta\}T}
$$

where $\Delta$ has arity $\alpha$.

**Lemma A.6** If (Func$_\alpha$) is admissible, so is

$$
(KFunc_\alpha) \quad \frac{\Gamma, \Delta, \Theta \Vdash K \text{ kind} \quad \Gamma \Vdash \vec{F} = \vec{G} : \Delta}{\Gamma, \{\vec{F}/\Delta\}\Theta \Vdash \{\vec{F}/\Delta\}K = \{\vec{G}/\Delta\}K}
$$

where $\Delta$ has arity $\alpha$.

**Proof** The proof is by induction on $K$. If $K \equiv \text{Type}$, there is nothing to prove; if $K$ has the form $\text{El}(A)$, we simply apply (Func$_\alpha$). If $K \equiv (y : K_1)K_2$, then the first premise is $\Gamma, \Delta, \Theta, y : K_1 \Vdash K_2$ kind, from which we obtain $\Gamma, \Delta, \Theta \Vdash K_1$ kind by Context Validity; apply the induction hypothesis to both of these.

**Lemma A.7** Let $\alpha$ be an arity. Suppose that every member of $R(\beta)$ is admissible for every proper subarity $\beta$ of $\alpha$. Then so are (Func$_\alpha$) and

$$
(RFunc_\alpha) \quad \frac{\Gamma, \Delta, \Theta \vdash M : T \quad \Gamma \Vdash \vec{F} = \vec{G} : \Delta}{\Gamma, \{\vec{F}/\Delta\}\Theta \vdash \{\vec{G}/\Delta\}M : \{\vec{F}/\Delta\}T}
$$

where $\Delta$ has arity $\alpha$. 
Lemma A.8 Let \( \alpha \) be an arity. Suppose every member of \( \mathcal{R}(\beta) \) is admissible for every subarity \( \beta \) of \( \alpha \). Let \( K \) be a kind of arity \( \alpha \). Suppose:

- every member of \( \Gamma, x : K, \Delta \vdash \bar{H} :: \Theta \) is \( \alpha \)-good and derivable;
- \( \Gamma, \{ F/x \} \Delta, \{ F/x \} \Theta \) valid; \( \Gamma \vdash F = G : K \); \( \Gamma \vdash F : K \); \( \Gamma \vdash G : K \) are all derivable.

Then the following are derivable:

\[
\begin{align*}
\Gamma &\vdash \{ F/x \} \Delta, \{ F/x \} \bar{H} = \{ G/x \} \bar{H} :: \{ F/x \} \Theta \\
\Gamma &\vdash \{ F/x \} \Delta, \{ G/x \} \bar{H} :: \{ F/x \} \Theta
\end{align*}
\]

Proof The proof is by induction on the length \( \bar{H} \) and \( \Theta \).

If they are of length 0, we apply (cut\( \alpha \)).

If they are of length 1, let

\[ \Theta \equiv y : (\Phi)T, \quad \bar{H} \equiv [\text{dom } \Phi]M. \]

(\( \approx \) In TF\( k \), \( \bar{H} \) would have the form \([\Phi']M\) for some context \( \Phi' \), but the following statements still hold.) We are given that \( \Gamma, x : K, \Delta, \Phi \vdash M : T \) is \( \alpha \)-good and derivable. It follows by (func\( \alpha \)) and (Rfunc\( \alpha \)) that

\[
\begin{align*}
\Gamma &\vdash \{ F/x \} \Delta, \{ F/x \} \Phi \vdash \{ F/x \} M = \{ G/x \} M : \{ F/x \} T \\
\Gamma &\vdash \{ F/x \} \Delta, \{ G/x \} \Phi \vdash \{ G/x \} M : \{ F/x \} T
\end{align*}
\]

36
as required.

Suppose they are of length \( m + 1 \), where \( m > 0 \). Let

\[
\vec{H} \equiv \vec{H}_0, \; H' \; ; \quad \Theta \equiv \Theta_0, \; y : L .
\]

The induction hypothesis gives us

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{F/x\} \vec{H}_0 = \{G/x\} \vec{H}_0 :: \{F/x\} \Theta_0
\]

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{G/x\} \vec{H}_0 :: \{F/x\} \Theta_0
\]

Also, the case of length 1 above gives us

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{F/x\} \vec{H}_0 = \{G/x\} \vec{H}_0 :: \{F/x\} \Theta_0
\]

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{F/x\} \vec{H}_0 :: \{F/x\} \Theta_0
\]

It remains only to show

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{G/x\} \vec{H}' : \{F/x\} \vec{H}_0 \Theta_0 \{F/x\} \Theta_0
\]

We are given \( \Gamma, \{F/x\} \Delta, \{F/x\} \Theta_0 \vdash \{F/x\} L \) kind, hence

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{F/x\} \vec{H}_0 :: \{F/x\} \Theta_0 \quad \text{(Lemma A.5)}
\]

\[
\Gamma, \{F/x\} \Delta \quad \vdash \quad \{F/x\} \vec{H}_0 :: \{F/x\} \Theta_0 \quad \text{(Lemma A.6)}
\]

and the result follows by (conversion \( \alpha_{m+1} \)).

\[\square\]

**Lemma A.9** Let \( \alpha \) be an arity. Suppose every member of \( R(\beta) \) is admissible for every subarity \( \beta \) of \( \alpha \). Then so is

\[
\begin{align*}
\text{(Conv}_{\alpha} & ) \quad \Gamma \vdash \vec{F} :: \Delta \quad \implies \quad \Gamma \vdash \vec{F} :: \Delta' \\
\end{align*}
\]

where \( \Delta \) has arity \( \alpha \).

**Proof** The proof is by induction on \( n \). In the case \( n = 0 \), there is nothing to prove. If \( n = m + 1 \), let

\[
\vec{F} \equiv \vec{F}_0, F'; \quad \Delta \equiv \Delta_0, x : K; \quad \Delta' \equiv \Delta_0, x : K' .
\]

The induction hypothesis gives \( \Gamma \vdash \vec{F} :: \Delta_0 \). We are also given \( \Gamma \vdash F' : \{\vec{F}/\Delta\} K \) and \( \Gamma, \Delta \vdash K = K' \). By Lemma A.5

\[
\Gamma \vdash \{\vec{F}/\Delta\} K = \{\vec{F}/\Delta\} K'
\]

and the result follows by (conversion \( \alpha_{m+1} \)).

\[\square\]
Lemma A.10 Let $\alpha$ be an arity. Suppose every member of $\mathcal{R}(\beta)$ is admissible for every subarity $\beta$ of $\alpha$. Then so is

$$\frac{(\text{Conv}_\text{Eq}, \alpha) \quad \Gamma \vdash \vec{F} = \vec{G} :: \Delta \quad \Gamma \vdash \Delta = \Delta' \quad \Gamma \vdash \vec{F} :: \Delta}{\Gamma \vdash \vec{F} = \vec{G} :: \Delta}$$

where $\Delta$ has arity $\alpha$.

**Proof** Similar. \qed

Theorem A.11 For every arity $\alpha$, every derivable judgement $J$ is $\alpha$-good.

**Proof** The proof is by a double induction, firstly on the arity $\alpha$, secondly on the derivation of $J$. We deal with the four rules in $\mathcal{R}(\alpha)$ one by one, and give the details only for the most difficult cases.

$(\text{cut}, \alpha)$ Suppose that $J \equiv \Gamma, x : K, \Delta \vdash J$ and that $\Gamma \vdash F : K$ is derivable. We deal with the case where the last step in the derivation of $J$ was

$$\frac{\Gamma, x : (\Theta)T, \Delta \vdash \vec{H} = \vec{H}' :: \Theta \quad \Gamma, x : (\Theta)T, \Delta \vdash \vec{H} :: \Theta \quad \Gamma, x : (\Theta)T, \Delta \vdash \vec{H}' :: \Theta}{\Gamma, x : (\Theta)T, \Delta \vdash x\vec{H} = x\vec{H}' : \{\vec{H}/\Theta\}T}$$

where $K \equiv (\Theta)T$. Let $F \equiv [\text{dom } \Theta]M$ and $G \equiv [\text{dom } \Theta']M$. Then the induction hypothesis gives

$$\Gamma, \{F/x\} \Delta \vdash \{F/x\} \vec{H} = \{F/x\} \vec{H}' :: \Theta$$

$$\Gamma, \{F/x\} \Delta \vdash \{F/x\} \vec{H} :: \Theta$$

$$\Gamma, \{F/x\} \Delta \vdash \{F/x\} \vec{H}' :: \Theta$$

Further, we are given $\Gamma, \Theta \vdash M : T$, and so by Lemma A.7

$$\Gamma, \{F/x\} \Delta \vdash \{\{F/x\} \vec{H}/\Theta\}M = \{\{F/x\} \vec{H}'/\Theta\}M : T$$

as required.

$(\text{func}, \alpha)$ Suppose that $J \equiv \Gamma, x : K, \Delta \vdash J$ and that $\Gamma \vdash F = G : K$, $\Gamma \vdash F : K$ and $\Gamma \vdash G : K$ are all derivable. We deal with the case where the last step in the derivation of $J$ was

$$\frac{\Gamma, x : (\Theta)T, \Delta \vdash \vec{H} :: \Theta}{\Gamma, x : (\Theta)T, \Delta \vdash x\vec{H} : \{\vec{H}/\Theta\}T}$$

where $K \equiv (\Theta)T$. Let $F \equiv [\text{dom } \Theta]M$ and $G \equiv [\text{dom } \Theta]N$ (§or $F \equiv [\Theta_1]M$ and $G \equiv [\Theta_2]N$). Then the induction hypothesis and Lemma A.8 give

$$\Gamma, \{F/x\} \Delta \vdash \{F/x\} \vec{H} = \{G/x\} \vec{H} :: \Theta$$

$$\Gamma, \{F/x\} \Delta \vdash \{F/x\} \vec{H} :: \Theta$$

$$\Gamma, \{F/x\} \Delta \vdash \{G/x\} \vec{H} :: \Theta$$

38
We are also given
\[ \Gamma, \Theta \vdash M = N : T; \quad \Gamma, \Theta \vdash M : T; \quad \Gamma, \Theta \vdash N : T. \]

Therefore,
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H}/\Theta \cdot M = \{F/x\} \bar{H}/\Theta \cdot N : \{F/x\} \bar{H}/\Theta \cdot T \tag{Lemma A.5} \]
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H}/\Theta \cdot N = \{G/x\} \bar{H}/\Theta \cdot N : \{F/x\} \bar{H}/\Theta \cdot T \tag{Lemma A.6} \]
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H}/\Theta \cdot M = \{G/x\} \bar{H}/\Theta \cdot N : \{F/x\} \bar{H}/\Theta \cdot T \tag{trans} \]
as required.

\textbf{(Rfunc)} Suppose that \( J \equiv \Gamma, x : K, \Delta \vdash M : T \) and that \( \Gamma \vdash F = G : K \), \( \Gamma \vdash F : K \) and \( \Gamma \vdash G : K \) are all derivable.

We deal with the case where the last step in the derivation of \( J \) was
\[ (\text{var}) \quad \Gamma, x : K, \Delta \vdash \bar{H} :: \Theta \]
\[ \Gamma, x : K, \Delta \vdash y \cdot \bar{H} : \{\bar{H}/\Theta\} \cdot T \]
where \( y : (\Theta) T \) is in \( \Gamma \) or \( \Delta \).

By the induction hypothesis and Lemma A.8
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H} = \{G/x\} \bar{H} : \{F/x\} \Theta \]
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H} :: \{F/x\} \Theta \]
\[ \Gamma, \{F/x\} \Delta \vdash \{G/x\} \bar{H} :: \{F/x\} \Theta \]

Therefore, applying (var) gives
\[ \Gamma, \{F/x\} \Delta \vdash \{G/x\} \bar{H} : \{G/x\} \bar{H}/\Theta \cdot \{F/x\} \cdot T. \]

It remains to show
\[ \Gamma, \{F/x\} \Delta \vdash \{F/x\} \bar{H}/\Theta \cdot \{F/x\} \cdot T = \{G/x\} \bar{H}/\Theta \cdot \{F/x\} \cdot T \tag{7} \]
after which the result will follow by (sym) and (conv). Context Validity gives
\[ \Gamma, \{F/x\} \Delta, \{F/x\} \Theta \vdash \{F/x\} \cdot T \text{ kind} \]
from which \( \text{7} \) follows by Lemma A.6.

\textbf{(context_conversion)} Suppose that \( J \equiv \Gamma, x : K, \Delta \vdash J \) and that \( \Gamma \vdash K = K' \) is derivable.

We deal with the case where the last step in the derivation of \( J \) was
\[ (\text{var_eq}) \quad \Gamma, x : (\Theta) T, \Delta \vdash \bar{F} = \bar{G} : \Theta \quad \Gamma, x : (\Theta) T, \Delta \vdash \bar{G} : \Theta \]
\[ \Gamma, x : (\Theta) T, \Delta \vdash x \cdot \bar{F} = x \cdot \bar{G} : \{\bar{F}/\Theta\} \cdot T \]

39
where $K \equiv (\Theta)T$. Let $K' \equiv (\Theta')T'$. By the induction hypothesis,

$$
\Gamma, x : (\Theta')T', \Delta \vdash \vec{F} = \vec{G} :: \Theta' \\
\Gamma, x : (\Theta')T', \Delta \vdash \vec{F} :: \Theta' \\
\Gamma, x : (\Theta')T', \Delta \vdash \vec{G} :: \Theta'
$$

We are given that $\Gamma \vdash \Theta = \Theta'$ and $\Gamma, \Theta \vdash T = T'$. We therefore have

$$
\Gamma, x : (\Theta')T', \Delta \vdash \vec{F} = \vec{G} :: \Theta' \quad \text{(Lemma A.10)}
$$

$$
\Gamma, x : (\Theta')T', \Delta \vdash \vec{F} :: \Theta' \quad \text{(Lemma A.9)}
$$

$$
\Gamma, x : (\Theta')T', \Delta \vdash \vec{G} :: \Theta' \quad \text{(Lemma A.9)}
$$

$$
\Gamma, x : (\Theta')T', \Delta \vdash x\vec{F} = x\vec{G} : \{\vec{F}/\Theta\}T' \quad \text{(var_eq')}
$$

$$
\Gamma, x : (\Theta')T', \Delta \vdash \{\vec{F}/\Theta\}T = \{\vec{F}/\Theta\}T' \quad \text{(Lemma A.5)}
$$

$$
\Gamma, x : (\Theta')T', \Delta \vdash x\vec{F} = x\vec{G} : \{\vec{F}/\Theta\}T \quad \text{(sym, conv)}
$$

as required. □

Now that this large proof has been completed, we are able to show that the following property, which we call Equation Validity, holds for TF'. It is then a short step to showing that TF and TF' are equivalent, and hence that all the properties we have proved for TF' also hold for TF.

**Theorem A.12 (Equation Validity)** In TF' or TF'_k, suppose that $\Gamma \vdash M = N : T$. Then $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$.

**Proof** The proof is by induction on the derivation of $\Gamma \vdash M = N : T$. We give the details for the case where the last step in the derivation was

$$
\Gamma \vdash \vec{F} = \vec{G} :: \Delta \\
(x : (\Delta)T \in \Gamma)
$$

Applying (var) to the second and third premises gives

$$
\Gamma \vdash x\vec{F} : \{\vec{F}/\Delta\}T \quad \Gamma \vdash x\vec{G} : \{\vec{G}/\Delta\}T.
$$

Lemma [A.6] gives us $\Gamma \vdash \{\vec{F}/\Delta\}T = \{\vec{G}/\Delta\}T$; the result follows. □

**Corollary A.12.1** In TF' or TF'_k, suppose that $\Gamma \vdash F = G : K$. Then $\Gamma \vdash F : K$ and $\Gamma \vdash G : K$.

**Corollary A.12.2** In TF' or TF'_k, suppose that $\Gamma \vdash \vec{F} = \vec{G} :: \Delta$. Then $\Gamma \vdash \vec{F} :: \Delta$ and $\Gamma \vdash \vec{G} :: \Delta$.

**Proof** The proof is by induction on the length of $\vec{F}, \vec{G}$ and $\Delta$. If the length is 0, there is nothing to prove.

Otherwise, let $\vec{F} \equiv \vec{F}_0, F'; \vec{G} \equiv \vec{G}_0, G'$; and $\Delta \equiv \Delta_0, x : K$. The induction hypothesis shows $\Gamma \vdash \vec{F}_0 :: \Delta_0$ and $\Gamma \vdash \vec{G}_0 :: \Delta_0$; it remains to show

$$
\Gamma \vdash F' : \{\vec{F}_0/\Delta_0\}K, \quad \Gamma \vdash G' : \{\vec{G}_0/\Delta_0\}K.
$$
We are given that $\Gamma \vdash F' = G' : \{\vec{F}_0/\Delta_0\}K$. By the previous corollary, it follows that

$$\Gamma \vdash F' : \{\vec{F}_0/\Delta_0\}K, \quad \Gamma \vdash G' : \{\vec{F}_0/\Delta_0\}K.$$  

Lemma A.6 gives us that $\Gamma \vdash \{\vec{F}_0/\Delta_0\}K = \{\vec{G}_0/\Delta_0\}K$, and the result follows.

\[\square\]

**Corollary A.12.3** The systems $TF$ and $TF'$ have the same derivable judgments. The systems $TF_k$ and $TF'_k$ have the same derivable judgments.

**Proof** The previous corollary shows that the new premises in (var\_eq') and (const\_eq') are redundant. \[\square\]

**Corollary A.12.4** The following properties hold in $TF$ and $TF_k$:

1. **Cut**
   
   If $\Gamma, x : K, \Delta \vdash J$ and $\Gamma \vdash F : K$ then $\Gamma, \{F/x\} \Delta \vdash \{F/x\}J$.

2. **Functionality**
   
   If $\Gamma, x : K, \Delta \vdash M : T$ and $\Gamma \vdash F = G : K$ then $\Gamma, \{F/x\} \Delta \vdash \{F/x\}M = \{G/x\}M : \{F/x\}T$.

3. **Context Conversion**
   
   If $\Gamma, x : K, \Delta \vdash J$ and $\Gamma \vdash K = K' \text{ then } \Gamma, x : K', \Delta \vdash J$.

4. **Equation Validity**
   
   If $\Gamma \vdash M = N : T$ then $\Gamma \vdash M : T$ and $\Gamma \vdash N : T$. 